Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Antti Karjakin

Silverlight-teknologia ohjelmistokehityksessä
Case: Ryhti™

Insinöörityö 24.6.2010

Ohjaaja: ohjelmistokehityspäällikkö Mika Ruottinen
Ohjaava opettaja: yliopettaja Kirsti Äystö

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä Otsikko Sivumäärä Aika	Antti Karjakin Silverlight-teknologia ohjelmistokehityksessä Case: Ryhti™ 75 sivua 24.6.2010
Koulutusohjelma	tietotekniikka
Tutkinto	insinööri (AMK)
Ohjaaja Ohjaava opettaja	ohjelmistokehityspäällikkö Mika Ruottinen yliopettaja Kirsti Äystö

Internet-sovellukset ovat olleet käytössä internetin syntyhetkistä lähtien. Kotioloissa tutuimmat internet-sovellukset ovat erilaiset pankkisovellukset.

Insinöörityö sai alkusysäyksen kun Olof Granlund Oy:ssä oltiin kiinnostuneita ottamaan Microsoft Silverlight mukaan kehitystyöhön. Olof Granlund Oy kehittää ja ylläpitää omaa internet-sovellusta nimeltä Ryhti™. Ryhdissä on käyttäjän nähtävillä yhteenvetosivu erilaisista prosesseista. Yhteenvetosivua varten tutkittiin Silverlightin mahdollisuuksia, rajoituksia ja käyttöönottoa.

Insinöörityön tavoitteeksi asetettiin Silverlight-teknologialla toteutettu prototyyppi. Prototyypin tulee tarjota vaihtoehtoinen ratkaisu yhteenvetosivun toteutukselle, joka on tehty perinteisemmillä internet-tekniikoilla. Prototyypin toteutuksen ohella tuli huolehtia, että Silverlightin käytön kaikki erityishuomiot tulee kirjatuksi muistiin.

Prototyyppi valmistui asetetussa aikataulussa ja sen tarjoamiin uusin ulottovuuksiin ollaan hyvin tyytyväisiä. Prototyyppiä koekäyttäneet ovat myös olleet hyvin kiinnostuneita Silverlightin tarjoamasta lisäarvosta käyttäjäkokemukseen.

Työn tuloksena syntyi myös dokumentti, johon on kerättynä huomioita, rajoituksia ja neuvoja Silverlightin käytöstä sovelluskehityksestä.

Hakusanat	Microsoft Silverlight, RIA, ohjelmistokehitys, HTML, ASP.Net

Helsinki Metropolia University of Applied Sciences **Abstract**

Author Title Number of Pages Date	Antti Karjakin Silverlight technology in software development Case: Ryhti™ 75 24 June 2010
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor Supervisor	Mika Ruottinen, Software Development Manager Kirsti Äystö, Principal Lecturer

Internet applications have been available since the Internet was born. The most common Internet applications that home users know are some sort of banking applications.

This thesis was commissioned when there were thoughts for using Silverlight technology in an Internet application at Olof Granlund Oy. Olof Granlund Oy develops and manages an Internet application called Ryhti™. Ryhti has a special summary page that involves different process details. The main objective for this thesis was to produce an alternative solution for building this page.

The objective of this thesis was to develop a prototype. That presents a summary page for the Internet application. Page had to be developed using Microsoft Silverlight technology. And also special observations were to be recorded.

The prototype was finished in given time and it received a very warm welcome in the company. Test users who had seen and tested the prototype had been very interested about Silverlight technology and richer user experience than a prototype is giving to a user.

Also as the outcome of this thesis a document was produced that has a collection of observations, restrictions and advice for using Silverlight in software development.

Keywords	Microsoft Silverlight, RIA, software development, HTML, ASP.Net

Lyhenne- ja käsitteluettelo

Apache	Apache HTTP Server on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma.
API	Application Programming Interface, ohjelmointirajapinta.
BCL	Base Class Libraries. Perusluokkien kirjasto, joka on saatavilla jokaiseen ohjelmointikieleen joka käyttää .Net ajoympäristöä.
BETA	Ohjelmiston esiversio, jossa on ohjelmistoon tarkoitetut ominaisuudet pääsääntöisesti valmiina.
C#	C sharp, on Microsoftin .Net-konseptia varten kehittämä ohjelmointikieli.
CERN	Conseil Européen pour la Recherche Nucléaire, Euroopan hiukkasfysiikan tutkimuslaitos.
CLR	Common Language Runtime, .NET-konseptin pääkomponentti.
DLR	Dynamic Language Runtime, CLR:n laajennus, joka tarjoaa palveluita dynaamisille ohjelmointikielille.
HLSL	High Level Shader Language, Microsoftin kehittämä kieli heijastustekniikan toteuttamiseen Direct 3D:llä.
HTML	Hypertext Markup Language. Avoimesti standardoitu kuvauskieli, joka tunnetaan erityisesti web-sivujen ansioista.
http	Hypertext Transfer Protocol, sovelluskehyksen protokolla tiedon välittämiseen yleisesti käyttäjän ja palvelimen koneen välillä.
Huijaus	Luodaan ohjelma jonka ulkonäkö on mahdollisimman tarkka kopio jo olemassa olevasta sovelluksesta ja yritetään näin huijata mahdollisia uhreja luovuttamaan esimerkiksi salasanatietoja.
IIS	Internet Information Services (tai Server). Microsoftin kehittämä palvelinohjelmistokokonaisuus.
JSON	JavaScript Object Notation. Avoimeen standardiin ja Java Scriptiin pohjautuva, tekstipohjainen tapa välittää tietoa.
LightTPD	LightTPD on nopeuskriittisiin ympäristöihin suunniteltu palvelinohjelmisto. LightTPD on käytössä useilla suurilla internet-sivustoilla, kuten YouTubessa, Wikipediassa ja meebossa.

LINQ — Language Integrated Query, Microsoftin .NET-ajoympäristön komponentti, jonka avulla mahdollistetaan tietokantakyselyt.

MIME — Multipurpose Internet Mail Extensions, on internet-teknologian standardi, joka kertoo sisällön tiedostomuodon.

Proseduraalinen ohjelmointi

Ohjelmointitapa, jossa ohjelman suoritus toteutetaan jakamalla ohjelman aliohjelmiin. Aliohjelmat ovat itsenäisiä ohjelman osia, joita voidaan kutsua mistä tahansa pääohjelmasta tai vaihtoehtoisesti muista aliohjelmista.

RIA — Rich Internet Application. Rikas Internet-sovellus on internet-sovellus, jossa on työpöytäsovelluksesta tuttuja elementtejä.

Sarjallistuminen

Sarjallistuminen on ohjelmointiluokan käyttämän tiedon muuntamista biteiksi ja tämän jälkeen bittien tallentaminen tietovarastoon (esimerkiksi tiedostoon). Sarjallistuvan luokan ominaisuuksiin kuuluu myös tiedon palauttaminen biteistä takaisin.

W3C — World Wide Web Consortium, kansainvälinen yritysten ja yhteisöjen yhteenliittymä, joka ylläpitää ja kehittää WWW-standardeja ja suosituksia.

WCF — Windows Community Foundation, API jolla .NET-ajoympäristössä kehitetään palvelupohjaisia sovelluksia.

WPF — Windows Presentation Foundain, on graafinen komponentti, käyttöliittymien renderöintiin Windows-pohjaisissa sovelluksissa.

XAML — Extensible Application Markup Language, on XML:ään pohjautuva, Microsoftin kehittämä kuvauskieli, jolla kuvataan käyttöliittymien rakennetta.

XML — Extensible Markup Language, on standardi, jolla dokumentissa olevaa tietoa voidaan kuvata, hyvänä esimerkkinä nykypäivän rss-syötteet.

WWW — World Wide Web, yleisesti puhuttaessa tarkoitetaan internetiä, virallisesti kyseessä on järjestelmä, joka koostuu yhteenlinkitetyistä dokumenteista.

WYSIWYG — What You See Is What You Get. Se mitä näet, on se mitä saat. Yleisesti puhutaan editorista, jonka luontivaiheen luomus vastaa loppuvaihetta, esimerkiksi tekstinkäsittely- ja esitysgrafiikkaohjelmat.

Sisällys

1 Johdanto

Internet on mullistanut valtavasti jokaisen ihmisen normaalia elämää. Miltä kuulostaisi työskentely tai pankkiasiointi täysin ilman tiedon valtaväylää internetiä? Internetissä toimivat palvelut ovat kuitenkin kehityksessä olleet aina jäljessä työpöytäsovelluksia, niin toiminnallisuudessa kuin interaktiivisuudessa.

Dynaamiset internet-sivut toivat aikoinaan suuren mullistuksen internetin maailmaan. Työpöytäsovelluksista tutut toiminnallisuudet voisivat olla seuraava iso askel, jonka avulla käyttäjät alkavat vaatia enemmän myös puhtaasti internetissä toimivilta sovelluksilta.

Insinöörityössäni tutustutaan Microsoftin Silverlight-teknologiaan. Microsoft Silverlight on Adobe Flashin kaltainen ajoympäristö internet-sovellusten tueksi. Silverlight-teknologia kehittyy kovaa vauhtia ja on mullistamassa perinteisten internet-sovellusten maailmaa yhdistelemällä perinteiseen internet-sovellukseen elementtejä työpöytä-sovelluksista. Silverlight mahdollistaa myös avoimemmat rajat internet-sovelluskehitykselle haastamalla näin käyttöliittymäsuunnittelijat kuin myös sovellus-koodin kehittäjätkin.

Työn alkusysäys tulee insinööritoimisto Olof Granlund Oy:ltä. Granlund on Suomen johtava talotekniikan konsultti. Yrityksen päätoimialat ovat talotekninen suunnittelu, kiinteistönpidon konsultointi sekä suunnittelu- ja ylläpito-ohjelmistojen kehitys ja myynti. Granlund haluaa tarjota omien sovellusten käyttäjille jotain lisää, jotain ekstraa, jotain mitä muilta ei saa.

Työn painopiste on näin ollen uusien asioiden tutkimisessa ja uuden asian soveltami-sessa siinä, mitä Silverlight mahdollistaa, miten se toteutetaan ja mitä tämä kaikki vaatii käyttäjän sekä palvelimen tekniikalta ja millaisia rajoituksia Silverlight asettaa uutena tekniikkana. Työni tuloksena syntyy Silverlight-teknologiaan pohjautuva käyttöliittymä internet-sovellukselle.

Työn tavoitteena on oppia Silverlight-teknologian käyttö sekä saada aikaan prototyyppi, jolla voitaisiin toteuttaa vaihtoehtoinen käyttöliittymä perinteisen HTML-käyttöliittymän rinnalle. Käyttöliittymän suunnittelussa ja toteutuksessa pyritään luomaan uudenlaisia, tehokkaampia ja visuaalisempia käyttäjäkokemuksia.

2 Internet-sovellukset

Niin kauan kun internet on ollut jokaisen saatavilla, sinne on myös suunniteltu ja toteutettu erilaisia sovelluksia. Internet-sovellusten pohjana toimii HTML, jota luetaan ja tulkitaan internet-selainten avulla.

HTML ei yksinään mahdollista dynaamisia internet-sivuja, jotka ovat merkittävässä osassa, kun puhutaan internet-sovelluksista (sovelluksessa esitettyä tietoa tulee voida tarpeen vaatiessa lisäämään, muokkaamaan ja poistamaan). Tätä varten kehitettiin nk. palvelinpuolen koodeja (ASP.NET, PHP jne.), joilla internet-sivuun muodostetaan tietoa käyttäjän pyytäessä sivua palvelimelta.

Seuraavaa kehitysaskelta ollaan pikkuhiljaa ottamassa internet-sovellusten osalta, kun markkinoilla on tullut erillisiä ajoympäristöjä. Erillisten ajoympäristöjen avulla muodostetaan nk. rikkaita internet-sovelluksia, sovelluksia, joissa on työpöytä-sovelluksista tuttuja elementtejä.

2.1 HTML

HTML on avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hypertekstiä. HTML on tunnettu erityisesti kielenä, josta internet-sivut rakentuvat. HTML:n historia alkaa vuodesta 1989 CERNissä, jossa fyysikko Tim Berners-Lee hahmotteli vaihtoehtoista tapaa esittää dokumentteja olemassa olevan kirjavan formaatin sijaan.

Kehityksen rinnalla syntyi myös yksinkertainen verkkoprotokolla http. CERN käynnisti ensimmäisen WWW-palvelimensa vuonna 1991. Idea sai maailmanlaatuista kiinnostusta ja monet ideasta kiinnostuneet liittyivät CERNin palvelimeen. 1994 perustettiin kansainvälinen yritysten ja yhteisöjen yhteenliittymä W3C.

HTML suunniteltiin alun perin kuvaamaan sivun rakennetta, eikä niinkään ulkoasua, mutta kirjoittajat halusivat paremmat mahdollisuudet vaikuttaa myös dokumentin ulkoasuun. Selainvalmistajat vastasivat haasteeseen esittelemällä HTML-määritykseen kuulumattomia elementtejä, jotka jouduttiin näin ollen ottamaan myöhemmin mukaan viralliseen HTML-määritykseen käytännön standardeina. (9.)

Viimeisin julkaistu standardi on HTML 4.01, joka on julkaistu 24.12.1999. Kehitystyötä tehdään edelleen ja seuraavaa standardia työstetään työnimellä HTML5.

Perinteisellä HTML-tekniikalla toimittaessa ensin luodaan HTML-dokumentti, joka viedään WWW-palvelimelle. Käyttäjä ottaa selaimella yhteyden palvelimeen ja selain pyytää HTML-dokumenttia. Palvelin lähettää dokumentin selaimelle, joka tulkitsee dokumentin ja piirtää (renderöi) sivun käyttäjän luettavaksi.

Yhdestä dokumentista voidaan viitata (linkki) toiseen ja näin ollen voidaan koostaa isompi kokonaisuus monesta pienemmästä palasesta. Kyseessä on tällöin myös täysin staattiset internet-sivut, joissa sisältö on ennalta määrättyä ja selain tekee kaiken työn.

Tämän tekniikan avulla tehdään yksinkertaisemmat internet-sivut. Mahdollisesti käytetään jotain WYSIWYG-editoria. Tekniikka tulee kuitenkin ymmärtää ja hallita, jotta voidaan siirtyä eteenpäin tekniikoihin, joissa hyödynnetään palvelinpään tai asiakaspään koodeja.

Hyvin yksinkertainen HTML-dokumentti löytyy liitteestä 3.

2.2 ASP.Net

ASP.Net on internet-sovelluksien kehys, joka on Microsoftin kehittämä ja markkinoima. Kehys mahdollistaa dynaamisten internet-sivujen, -sovelluksien ja -palveluiden kehityksen. Kehyksen ensimmäinen versio julkaistiin neljän vuoden kehitystyön tuloksena tammikuussa 2002 (versio 1.0). ASP.Net-teknologia on Microsoftin ASP-teknologian seuraaja.

.Net-sivut, joista käytetään yleisesti termiä ”web forms”, tunnistaa helpoiten tiedostopäätteestä ”.aspx”. Sivut sisältävät yleensä staattista (X)HTML:ää sekä ns. palvelinpuolen koodia. Palvelinpuolen koodilla mahdollistetaan sivujen dynaaminen sisältö. ASP.Netin tekniikan toiminta on vastaava kuin kilpailijoiden (PHP, JSP). Osa sivusta on näin ollen kirjoitettu puhtaasti HTML:nä, ja sopiviin kohtiin sivulle ladataan käytettävällä palvelinpuolenkoodilla esimerkiksi tietokannasta dataa, joka voidaan esittää käyttäjälle muun muassa taulukkomuodossa.

Microsoft suositteleekin, että tällaisia sivuja luotaessa varsinainen palvelinpuolen koodi sijoitetaan niin sanottuun ”Code-behind”-malliin. Tällöin varsinaisen .aspx-sivun rinnalla kehitetään .aspx.cs- tai .aspx.vb-tiedostoa (tiedoston nimien tulee olla sama kuin varsinaisen .aspx-sivun, esimerkiksi minunEsimerkki.aspx ja minunEsimerkki.aspx.cs).

Tiedoston viimeisin pääte kertoo käytettävän ohjelmointikielen, jossa .cs viittaa Microsoftin .NET-konseptia varten kehitetyyn C#:iin ja .vb viittaa Microsoftin Visual Basic .NETtiin. VB.NET on kehitetty Microsoftin Visual Basic-ohjelmointikielestä.

Tyypillistä Code-behind-toteutukselle on, että ohjelmoija kirjoittaa koodin vastaamaan erilaisia tapahtumia (events). Esimerkiksi sivun lataus valmistuu (Page_Loaded) tai sovelluksessa tapahtuu jotain käyttäjän painaessa nappia (OnButtonClick), sen sijaan että kirjoitettaisiin proseduraalinen ohjelma.

ASP.NETistä tuttu malli on havaittavissa myös Silverlightin puolella, jossa käyttöliittymä voidaan toteuttaa XAML:n avulla ja toiminnot sijoittaa Code-behindiin. Vaihtoehtoisesti voidaan myös luoda komponentteja Code-behindin puolella käytettävällä .NET-ohjelmointikielellä ja sijoittaa niitä käyttöliittymään.

2.3 Rikas internet-sovellus (RIA)

RIA (Rich Internet Application) eli rikas internet-sovellus on määritelmän mukaan internet-sovellus, jonka useimmat ominaisuudet ovat tuttuja työpöytäsovelluksista. Näin ollen rikkaat internet-sovellukset pyrkivät parantamaan käyttäjäkokemusta ja samalla pyrkivät tarjoamaan perinteistä internet-sovellusta kehittyneemmän käyttöliittymän.

Edistyneemmästä teknologiasta huolimatta, rikkaat internet-sovellukset suoritetaan yleensä internet-selaimessa. Osa tällaisista sovelluksista on mahdollista suorittaa myös erikseen käyttäjän koneelle asennettavassa ajoympäristössä. Silverlightin kohdalla tällöin puhutaan niin sanotusta ”Out of browser” -sovelluksesta.

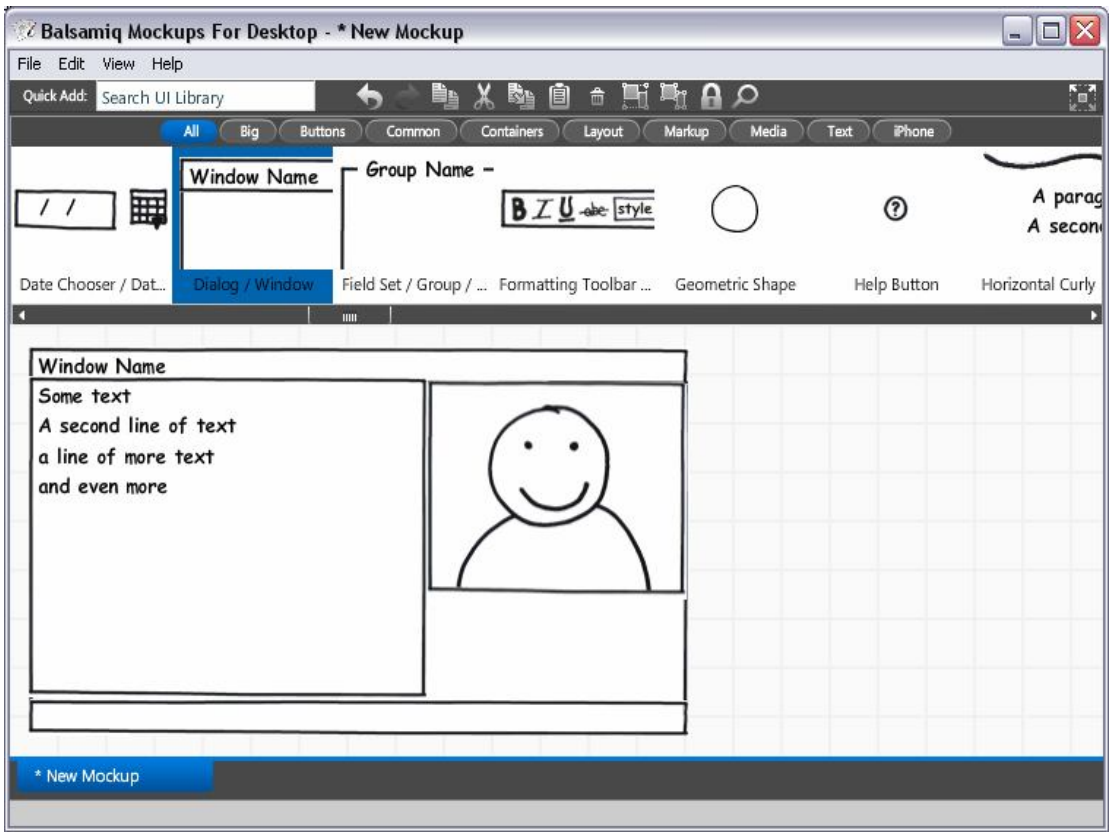
Rikkaassa internet-sovelluksessa käyttäjän ja sovelluksen välillä vallitsee kaksisuuntainen vuorovaikutus, sovelluksen tulee tarjota yhtä interaktiivisia komponentteja, kuin tavallinen työpöytäsovellus pystyy käyttäjälle tarjoamaan.

Käyttäjän tehdessä rikkaassa internet-sovelluksessa tapahtuman esimerkiksi napin painalluksen, sovellus vastaanottaa tämän, tekee tapahtumaa vastaavat toimensa ja antaa käyttäjälle palautteen takaisin. Esimerkiksi karttasovelluksessa käyttäjä pyytää sovellukselta jotain katuosoitetta. Ohjelman löytäessä kohteen se siirtää karttaikkunan pyydettyyn osoitteeseen tai vaihtoehtoisesti ilmoittaa käyttäjälle, että pyydettyä osoitetta ei löydy.

Tärkeimpänä kriteerinä rikkaissa internet-sovelluksissa on siirrettävän datan tehokkuus. Siirtoviiveet ja siirrettävä data tulee minimoida, jotta käyttäjä ei joudu turhaan pyörittelemään peukaloitaan odottaessaan sovellusta. Mitä sulavammin sovellus toimii, sitä paremman käyttäjäkokemuksen sovellus tarjoaa.

Liukusäätimien, vedettävien ja pudotettavien komponenttien sekä animoitujen siirtymien avulla on laajemmat mahdollisuudet käyttöliittymäsuunnitteluun, mikä parantaa käytettävyyttä ja antaa suuntaa positiiviselle käyttäjäkokemukselle. Lisäksi toiminnot tuovat internet-sovelluksia lähemmäksi työpöytäsovelluksia.

Tunnetuimpana esimerkkinä on Youtube, jossa internet-sovelluksen avulla käyttäjien videoita voidaan näyttää muille käyttäjille sekä Googlen karttapalvelu (Google Maps ja Google Earth). Näiden lisäksi myös Adoben AIR ajoympäristö, joka mahdollistaa vuorovaikutteisten internetsovellusten kehittämisen ja käyttämisen työpöytä-sovelluksina.



Kuva 1. Adobe AIR ja Balsamiq Mockups.

Yksi tehokas työkalu, joka on kehitetty Adoben AIR tekniikalla, on Balsamiq Mockups, joka mahdollistaa käyttöliittymien piirtämisen minuuteissa. Käyttöliittymiin voidaan lisätä myös hieman toiminnallisuutta ja näin niistä saadaan aikaan kevyitä esittelysovelluksia. Kuvasta 1 voi havaita yksinkertaisen Chat-ohjelman käyttöliittymän toteutettuna ko. sovelluksella.

3 Silverlight-teknologian kehityskaari

Microsoft Silverlight on internet-sovelluskehys, joka mahdollistaa samankaltaista toiminnallisuutta kuin Adoben Flash, integroitua multimediaa, grafiikkaa, animaatioita ja interaktiviisuutta yhdessä ajoympäristössä (1).

Microsoft Silverlight on yhteensopiva monien internet-selaimien kanssa, riippumatta pyörivätkö ne Windows-, Linux- tai Mac OS X-käyttöjärjestelmien päällä. Silverlightista on pyritty näin ollen kehittämään mahdollisimmat alustariippumaton ympäristö.

Internet-selaimiin ladattavasta Silverlight-liitännäisestä on myös pyritty tekemään mahdollisimman pienikokoinen, jotta sen lataaminen ja asentaminen tapahtuisi mahdollisimman nopeasti.

Silverlightin avulla voidaan lisäksi tuottaa liitännäisiä Vistan myötä esiteltyyn Windowsin sivupalkkiin.

3.1 Silverlight 1.0

Silverlight 1.0:tä kehitettiin aluksi työnimellä Windows Presentation Foundation/Everywhere (WPF/E). Ensimmäisen version virallinen julkistaminen ja markkinoille julkaisu tapahtui viides syyskuuta 2007.

Ominaisuuksien puolesta Silverlight 1.0 soveltui lähinnä suoratoisto- ja videosovelluksien kehitykseen. Silverlight 1.0 sovellus käynnistettiin kutsumalla XAML-tiedostoa HTML-sivulta. Varsinaisessa Silverlight-sovelluksessa oli pohjana Canvas-elementti, johon muut elementit sijoitettiin. Yksi isoista kehittäjien kohtaamista ongelmista oli dynaamisen sisällön hallittavuudella, koska ko. versiossa rajapinta oli mahdollistettu ainoastaan XML- tai JSON-datalle.

Silverlight 1.0 oli markkinoille saapuessaan hyvin rajoitettu ominaisuuksiltaan. Tuotteen ympärillä alkoi välittömästi kuhina käyttäjien, kehittäjien kuin kilpailevien yrityksen toimesta. Hyvin harva näki Silverlightilla minkäänlaista tulevaisuutta. Tarjosihan Adoben vastaava tuote paljon enemmän kuin uusi kilpailijansa tässä kohtaa.

3.2 Silverlight 2

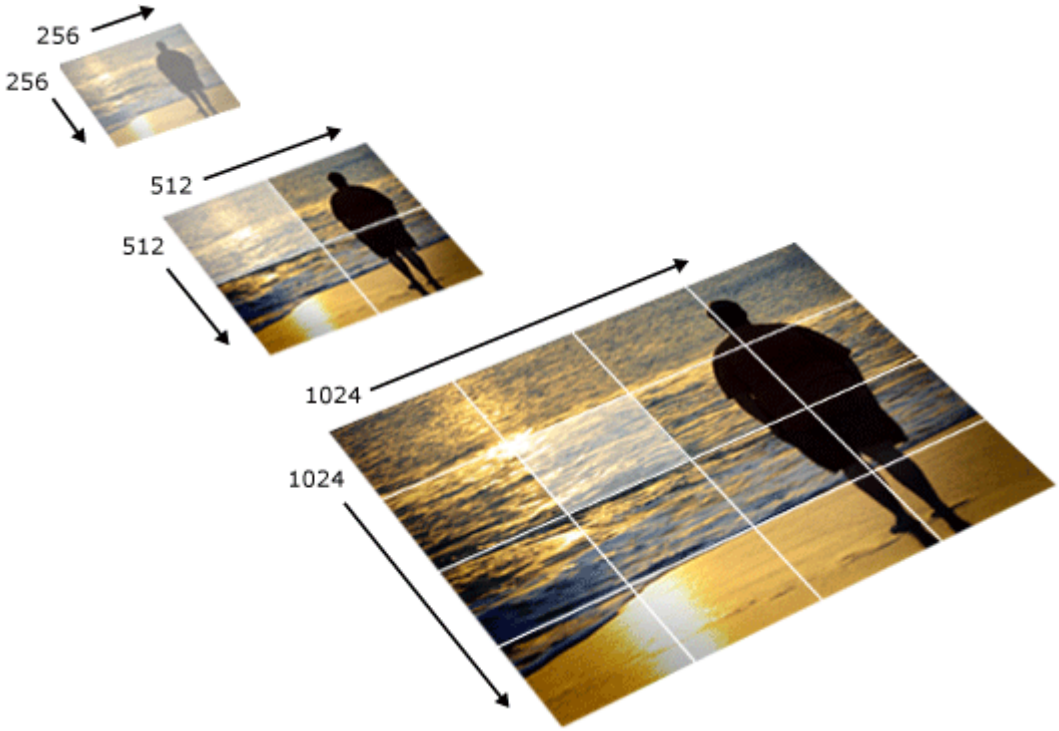
Silverlight 2 julkaistiin 14.10.2008. Julkaisu mullisti Silverlightin tarjoaman ajoympäristön käytettävyyden täysin. Moni sovelluskehittäjä alkoi nähdä Silverlightin aivan uudessa valossa tutustuessaan, mitä uusia ominaisuuksia Silverlight 2:n myötä oli tarjolla.

Silverlight 2 sisältää .NET Frameworkin, joka implementoi saman CLR-version kuin .NET Framework 3.0. Frameworkin myötä Silverlight-sovelluksia voidaan kirjoittaa kaikilla .NET-ohjelmointikielillä (VB, C#, JavaScript, IronPython ja IronRuby). Silverlight 2 toi kehittäjien ulottuvilla myös paljon rikkaita komponentteja, jotka olivat itsestäänselvyys internet-sovelluksissa. Kaikki komponentit tukevat template-mallia, näin ollen kehittäjällä on suoraviivainen malli komponentin lisäkehitykselle omiin tarpeisiin. (1.)

Silverlight 2:ssa sovellukset pakataan ZIP-tekniikalla XAP-tiedostoiksi. XAP-tiedosto sisältää ohjelman suoritukseen vaadittavat referenssit, lisäksi ohjelman käyttöliittymäpuolen XAML:n sekä ohjelmakoodin, joka on kirjoitettu Code-behindiin. Code-behind on termi, jota käytetään kuvaamaan koodia, joka liitetään yhteen varsinaisen XAML-koodin kanssa sovelluksen käännösvaiheessa.

Silverlight 2 mahdollistaa myös monisäikeisten ohjelmien kirjoittamisen.

Monessa Silverlight-esittelyssä näkyvä Deep Zoom toiminto esiteltiin suurelle yleisölle Silverlight 2 myötä. Kyseessä on tekniikka, jossa kuvat tuodaan ohjelmaan erillisellä ohjelmalla, joka lukee ja tulkitsee kuvat ja luo alkuperäisestä suuriresoluutiokuvasta nipun uusia kuvia, mikä mahdollistaa näin kuvan tarkkuuden säilymisen zoomattaessa.



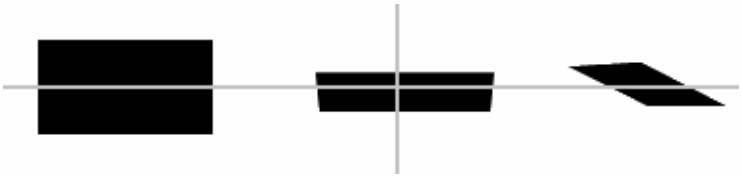
Kuva 2. Deep Zoom -työkalu paloittelee alkuperäisen suuriresoluutiokuvan. (14)

Deep Zoom -toiminnossa kuvaa tarjotaan käyttäjälle ensin pieniresoluutio kuvana. Tämän avulla suuren kuvan latausaika on pienempi kuin alkuperäisen suuriresoluutio kuvan. Käyttäjän zoomatessa kuvaa ladataan osa suuremman tarkkuuden kuvasta käyttäjän haluamasta kohtaa. Tämä mahdollistaa kuvan tarkkuuden säilymisen, kun käyttäjälle tuodaan haluttu kohta suuriresoluutioisesta kuvasta, kuin mistä zoomausta lähdettiin suorittamaan. Toiminnallisuus on havainnollistettu kuvassa 2.

3.3 Silverlight 3

Ensimmäisen kerran kolmannesta Silverlightista vihjailtiin IBC 2008-tapahtumassa, joka järjestettiin Amsterdamissa syyskuun 12. päivänä. Maaliskuun 18. päivä 2009 Las Vegasissa järjestetyssä MIX09-tapahtumassa julkaistiin betaversio Silverlight 3:sta, ja samana iltana käyttäjien oli mahdollista ladata tämä betaversio ja aloittaa tuotteeseen tutustuminen. Lopullinen versio julkaistiin heinäkuun yhdeksäntenä 2009.

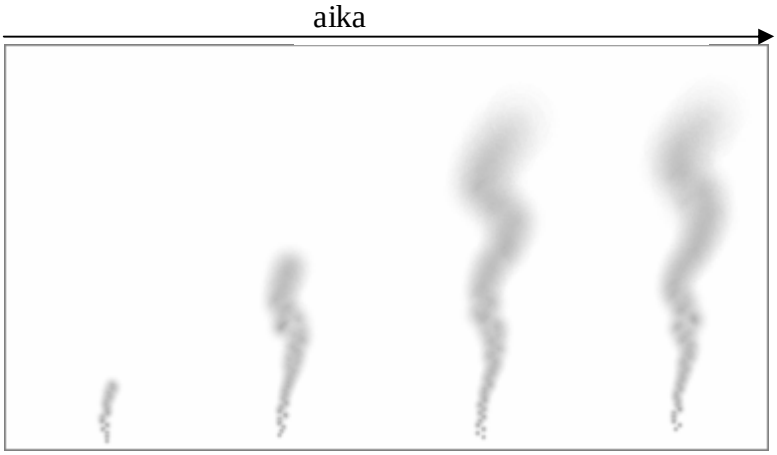
Silverlight 3:n myötä Silverlightin komponenttikirjasto kasvoi useilla uusilla komponenteilla. Nyt Silverlight omasi itsessään muun muassa puukontrollin (TreeView) ja datataulukon (DataGrid). Lisäksi Microsoft julkaisi erillisen Silverlight Toolkit-kirjaston, joka laajensi Silverlight-maailmaan muun muassa erilaisia kaaviomalleja.



Kuva 3. Tavallinen neliökomponentti, seuraavassa tilassa se on altistettu 65 asteen kääntämiselle x-akselinsa suhteen, kolmannessa vaiheessa on lisätty vielä 65 astetta kääntöä y-akselin suhteen.

Silverlight 3 tukee perspektiivistä 3D:tä, mikä mahdollistaa 2D-komponenttien pyörittelyn, kääntelyn ja venyttämisen kolmiulotteisessa maailmassa. Tätä on havainnollistettu kuvassa 3. Nämä muunnokset, kuten myös AAC äänen ja H.264 videon dekoodaus, voidaan Silverlight 3:n myötä suorittaa laitteistokiihdytyksen tukemana. Laitteistokiihdytys mahdollistaa sulavan animaation ja videokuvan aina 1080p-laatuisen materiaalin toistossa.

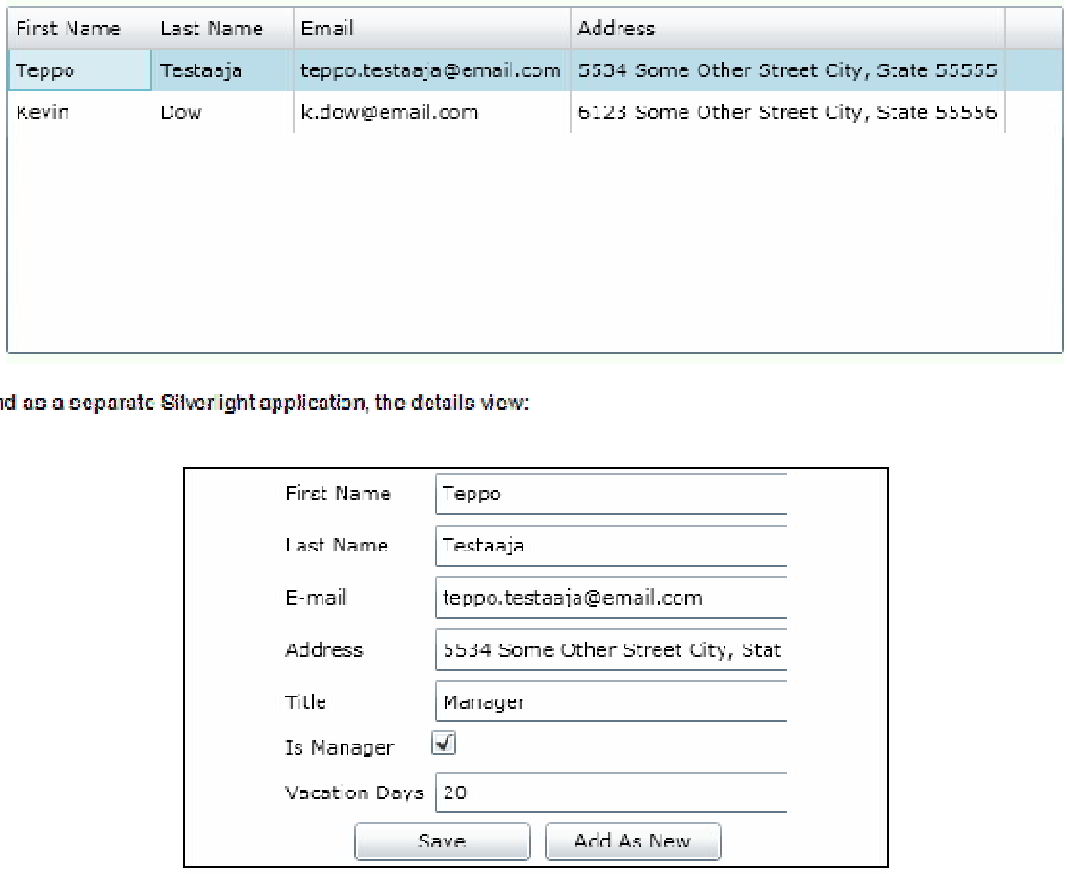
Komponenttien pyörittelyn ja kääntelyn lisäksi uutta Silverlight 3:ssa oli täysin kustomoitujen animaatioiden luonti käyttäen High Level Shader (HLSL) -kieltä, jonka avulla luodaan pikseliheijastuksia (pixel shader). Bittikarttoja voidaan myös muokata erillisellä API:lla. Tämä mahdollistaa kuvien muokkauksen ”lennosta”. Esimerkiksi voidaan luoda ohjelma, joka poistaa siihen ladattavista kuvista automaattisesti punasilmäisyyttä.



Kuva 4. Silverlightin avulla toteutettu pikselianimaatio.

Nämä kaikki kuvanmanipuloinnit vaatisivat prosessorilta paljon laskentatehoa. Silverlight 3 kehitettiin niin, että se kykenee hyödyntämään laitteiston grafiikkapiiriä eli näytönohjainta. Kaikki näkyvä on mahdollista myös ladata näytönohjaimen videomuistiin ja näin ollen varmistaa entistäkin sulavammat animaatiot.

Silverlight 3 oli myös ensimmäinen Silverlightin versio, jossa elementeistä voitiin sitoa dataa toiseen elementtiin, samaten sidotun tiedon tarkistaminen esiteltiin laajemmin Silverlight 3:ssa. Samalla tuotiin myös LocalConnection API, joka mahdollistaa kahden erillisen Silverlight-kontrollin tai sovelluksen kommunikoinnin keskenään. (20.)



Kuva 5. Silverlight Local Connection API -esimerkki HTML-sivulla. (15)

Kaksi erillistä Silverlight-elementtiä voidaan esimerkiksi sijoittaa HTML-sivun alkuun. Ensimmäisenä elementtinä on datataulukko (DataGrid), josta tieto on poimittavissa ja johon tieto tallennetaan. Jälkimmäisenä elementtinä on Silverlight-lomake, jolla syötetään ja muokataan tietoja. Tämänkaltainen esimerkki on esitetty kuvassa 5.

Toiminnallisuus helpottaa isojen sovelluskokonaisuuksien kehittämistä. Työryhmät voivat kehittää sovelluksiaan riippumatta toisistaan, kunhan rajapinnassa liikuteltavat yksityiskohdat ovat sovittuina.

Tiedostojen tallentaminen kehittyi myös Silverlight 3:ssa. Kun sovelluksesta tallennetaan esimerkiksi kaavion export, käyttäjä voi avautuvassa dialogissa navigoida mihin tahansa sijaintiin, joka hänelle näkyy. Silverlight-sovellus ei kuitenkaan tiedä, mihin käyttäjä tiedoston tallentaa, koska tätä polkua ei koskaan välitetä tietoturvasyistä Silverlightille. Aiemmassa versiossa käytössä olivat vain paikalliset tallennusvarastot.

Silverlight 3:ssa esiteltiin myös ”Out of browser” -toiminnallisuus eli sovelluksen toiminta selaimen ulkopuolella. Käyttäjä voi asentaa kopion sovelluksesta omalle kiintolevylle (ohjelmoijan tulee mahdollistaa tämä toiminnallisuus koodissa) ja käyttää sovellusta, vaikka kone ei olisi kytkettynä internetiin (olettaen että sovellus ei tarvitse internet-yhteyttä muuten). Ohjelman käynnistäminen tapahtuu pikakuvakkeesta käynnistä-valikosta tai vaihtoehtoisesti työpöydältä. (20.)

Sovelluksessa voidaan tarvittaessa tarkistaa, ajetaanko sovellusta selaimessa vai erillisestä asennuksesta. Vaikka Silverlight-sovellusta ajettaisiin omasta installaatiosta, se pyörii silti Silverlightin tarjoamassa ajoympäristössä. Käynnistettäessä tällaista paikallista asennusta Silverlight pyrkii tarkistamaan, onko sovellukseen tullut uudempaa versiota ja asentamaan uudemman version, jos sellainen on saatavilla.

Silverlight 3:lla voidaan suorittaa niin Silverlight 2:lle kuin Silverlight 1.0:lle kehitettyjä sovelluksia, mutta ei tietenkään päinvastoin. Tämä tulee huomioida, kun kehitetään aiemmilla versioilla kehitettyjä sovelluksia edelleen uudemmille versioille.

Jokaisella sovellusta käyttävällä ei välttämättä ole oikeuksia päivittää internet-selaimensa liitännäistä uudempaan. Mitä suuremmasta työympäristöstä on kysymys, sitä hankalampi käyttäjän on yleensä saada jokin lisäominaisuus vain omalle henkilökohtaiselle tietokoneelle.

Silverlight 3:n myötä ajoympäristö otti aimo harppauksen eteenpäin uusien komponenttien, toiminnallisuuksien ja käytettävyyden kannalta. Silverlight lähestyy kaiken aikaa kilpailijoitansa ja laajennettavien komponenttikirjastojen avulla se on jo mennyt osaltaan ohi.

XAP-tiedoston pakkausalgoritmia on parannettu paljon kakkosversioon verrattuna. Kakkosversiossa monet ohjeet ja neuvot puhuivat sen puolesta, että XAP-tiedosto kannattaisi nimetä uudestaan ZIP-tiedostoksi, purkaa sisältö zip-ohjelmalla johonkin kansioon ja tämän jälkeen pakata purettu tieto uudestaan varsinaisella zip-ohjelmalla.

Seuraavaksi ZIP-tiedosto muunnetaan takaisin XAP-tiedostoksi ja palvelimella käytetään tätä uudestaan pakattua XAP-tiedostoa. Näin ollen on mahdollista säästää paljonkin tilaa ja käyttäjälle välitettävän datan määrää, mutta algoritmin parantumisen myötä tällaisesta toiminnasta ei havaittu olevan enää kovin suurta hyötyä.

3.4 Silverlight 4 Beta

Silverlight 4 Beta on uusin saatavilla oleva versio Silverlightista (helmikuussa 2010). Vain neljä kuukautta edellisen version julkaisun jälkeen Microsoft paljasti Silverlight 4 Betan. Kehitystiimi on jatkanut tuotteen kehitystä kiivaasti ja Silverlight 4 Betan myötä Silverlight-ajoympäristö astui jälleen lähemmäksi kilpailijaansa.

Silverlight 4 Beta tarjoaa uusia rajapintoja ja mahdollisuuden kehittää sovelluksia yhä enemmän työpöytäsovellusten kaltaiseksi. Tämän mahdollistaa muun muassa tulostamisen tuen.

Tulostettaessa dokumenttien tai dynaamisia raportteja, uuden tuen avulla voidaan myös esikatsella dokumentteja. Lomakkeiden komponenttien lukumäärä kasvaa yli kuuteenkymmeneen muokattavaan ja stailattavaan (skinnable) komponenttiin.

Silverlight 3:sta tuttu selaimen ulkopuolella toimivan sovelluksen tuki saa uuden ominaisuuden. Tukea on laajennettu täyden luottamuksen ominaisuudella. Käyttäjän antaessa sovellukselle täyden luottamuksen (Full Trust) sovelluksesta voidaan tehdä käyttäjäystävällisempi. Täyden luottamuksen omaavassa sovelluksessa voidaan esimerkiksi kirjoittaa syöttökenttiin tekstiä, vaikka sovellus pyörisi kokoruudun tilassa.

Tiedoston avaus ja tallennustapahtumissa voidaan käyttää hyödyksi absoluuttisia polkuja, tämän lisäksi ko. dialogeja voidaan avata kokoruudun tilassa ilman käyttäjän vaatimia toimia.

Hiiritukea on laajennettu, tuen avulla on mahdollista käyttää myös hiiren kakkosnäppäintä ja rullan toimintaa on tarkennettu. Näin ollen ei DataGridiin tarvitse välttämättä ensimmäisenä koodata Automation-rajapintojen avulla hiiren rullan toiminnallisuutta sisällön vieritykseen.

Neljäs Silverlight-versio tarjoaa myös kokonaan uuden tavan välittää käyttäjälle viestejä ohjelmasta. Deep Zoomin sulavuutta on parannettu entisestään ja luotu kokonaan uusi tuki HTML:n esittämiselle (renderöinnille) Silverlightin sisällä.

4 Silverlightin turvallisuus

Silverlight-teknologiasta on pyritty tekemään mahdollisimman turvallinen käyttäjälle rajoittamalla toimintoja tietyissä tilanteissa ja kiinnitettämällä erityishuomiota muun muassa erilaisiin huijaustilanteisiin.

Erinomainen esimerkki Silverlight-sovelluksen turvallisuudesta on sen käynnistyminen. Kun selain HTML-sivulta kutsuu sovelluksen XAP-tiedostoa, avautuva Silverlight-sovellus pyörii omassa ”hiekkalaatikossaan” ja sillä voi olla erilaiset turvallisuus-rajoitukset kuin sitä ympäröivällä HTML-sivulla.

Oletusarvoisesti Silverlight-sovellus voi kommunikoida sitä esittävän HTML-sivun kanssa, jos HTML-sivu ja Silverlight-sovellus ovat saman verkkotunnuksen alla. Ohjelmistokehittäjä voi hallita kommunikointimahdollisuutta EnableHtmlAccess-attribuutin avulla (asettaa kommunikoinnin mahdolliseksi tai kieltää sen).

Ohjelmistokehittäjän tulee pitää mielessä, että käyttäjän koneelle ladataan koko XAP-tiedosto. Tämä tiedosto voidaan avata ja tiedostosta saatuja DLL-tiedostoja voidaan yrittää lukea erilaisilla kolmannen osapuolen sovelluksilla. Näin ollen jokainen joka voi avata sovelluksen, voi päästä käsiksi myös sovelluksen DLL-tiedostoihin. (18.)

Silverlight-sovelluksen kommunikoidessa käyttäjän tietokoneelta palvelimelle (esimerkiksi tilanteessa, jossa WCF-palvelu pyytää viimeisimpiä tietoja) käytetään yleisesti WSHttpBinding-luokkaa, joka käyttää tiedonsiirtoon http-protokollaa. Oletuksena tässä luokassa kommunikointia ei salata millään tavalla. Http-protokolla voidaan suojata Secure Sockets Layerillä (SSL), joka on yleisemmin tunnettu ja tunnistettu https:nä. (19.)

WSHttpBinding-luokka tukee SSL:n käyttöä. Ohjelmistokehittäjän vastuulla on vain sen asettaminen käyttöön, koska se ei ole oletuksena aktiivisena. Näin kaikki tieto, joka liikkuu käyttäjän ja palvelimen välillä, ei ole salaamatonta.

Silverlight voi tallentaa tietoja käyttäjän tietokoneelle käyttäjän tietämättä, mutta Silverlight ei pääse käsiksi käyttäjän tiedostojärjestelmään. Tallennus tapahtuu tässä tapauksessa aina paikalliseen tallennusvarastoon. Käyttäjä voi tietoturvaa tiukentaakseen estää kokonaan tiedon tallentamisen Silverlight-sovelluksesta.

Silverlight-sovellukseen voidaan viedä tiedostoja ja sovelluksesta voidaan tallentaa tiedostoja käyttäjän koneelle, mutta tämä tapahtuu aina erillisten valintaikkunan kautta, joten käyttäjä on kyllä tietoinen, mihin tietoa tallentaa tai mistä luovuttaa tietoja.

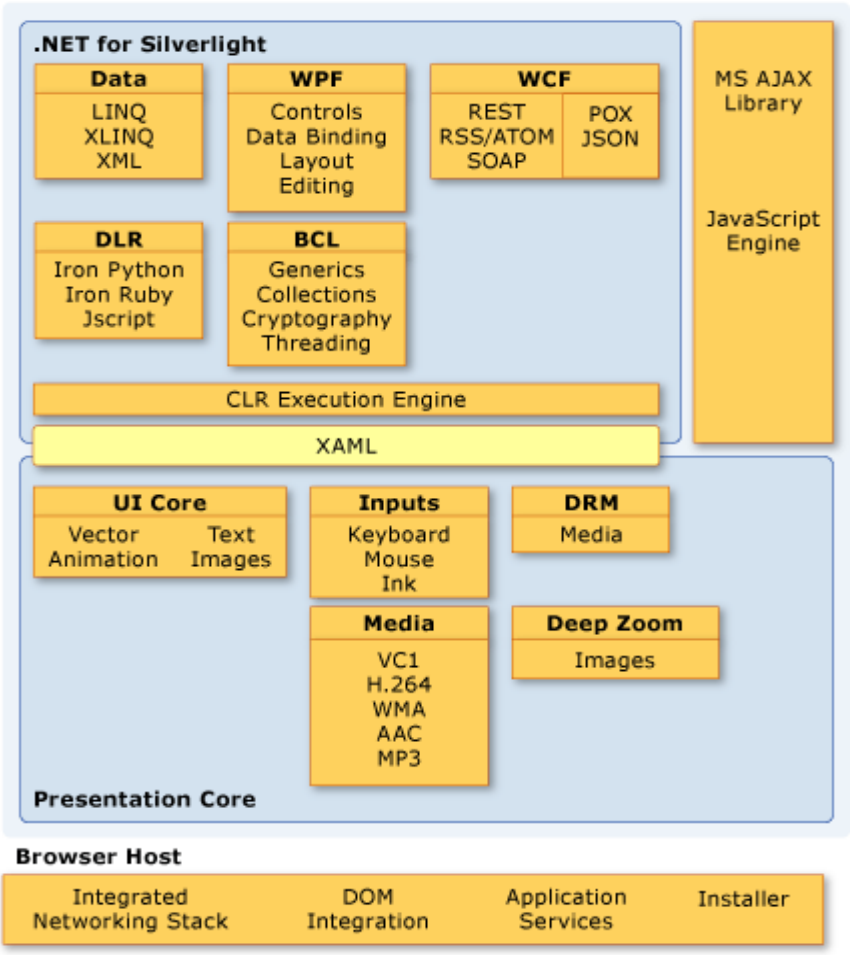
Käyttäjän näkökulmasta Silverlight on hyvin turvallinen teknologia, koska sen avulla ei voida suoraan tallentaa mitä tahansa käyttäjän koneelle. Käyttäjän täytyy aina olla valppaana, kun tuntemattomista sovelluksista halutaan tallentaa tietoa käyttäjän koneelle.

5 Silverlight-arkkitehtuuri

Silverlightin tarjoama ohjelmistokehys voidaan jakaa kahteen isompaan kokonaisuuteen, esityskehykseen ja Silverlightin .NET-kehykseen. Näiden kahden lisäksi arkkitehtuuri käsittää asennus- ja päivityskomponentit.

Asennus- ja päivityskomponentit tarjoavat nimensä mukaisesti Silverlightin asentamisen ensimmäistä kertaa käyttäjälle ja huomaamattoman ja helpon tavan Silverlightin päivittämiseen.

Sovellukselle voidaan määrittää tarvittavan Silverlight-liitännäisen minimiversio, jos käyttäjällä on tätä vanhempi versio, pyydetään käyttäjää päivittämään oma liitännäinen.



Kuva 6. Silverlight-arkkitehtuuri. (11)

5.1 Esityskehys (presentation core)

Esityskehyksessä sijaitsevat komponentit ja palvelut jotka toimivat ja vastaavat sovelluksen käyttöliittymän esittämisestä käyttäjälle. Käyttöliittymän ydin (UI Core) hahmontaa (renderöi) vektorit, bittikarttagrafiikan, huolehtii animoinneista ja tekstin esityksestä.

Tiedonsyöttöominaisuus (Inputs) vastaa käyttäjän tiedonsyötön välittämisestä Silverlightille. Ominaisuus on myös vastuussa näppäimistön painallusten, hiiren liikkeiden ja painallusten, sekä muiden tiedonsyötön välineiden syötön välityksestä Silverlight-sovellukselle.

DRM-ominaisuus mahdollistaa digitaalisten käyttöoikeuksien hallinnoinnin. Mediaominaisuudet vastaavat eri tyyppisten video- ja äänitiedostojen toistamisesta ja hallinnoinnista.

Deep Zoom-ominaisuus tarjoaa zoomaustoiminnot korkearesoluution kuville. Sijoitteluominaisuus (Layout) mahdollistaa käyttöliittymäkomponenttien dynaamisen sijoittelun. Esityskehyksessä kuvattu XAML ominaisuus huolehtii XAML:n jäsentämisestä. (11.)

5.2 Silverlightin .NET-sovelluskehys (.NET for Silverlight)

Kuvasta 6 voidaan nähdä, että .NETin tarjoama sovelluskehys Silverlightin tarpeisiin on jaoteltu kuuteen eri osaan. Toiset tarjoavat lisää visuaalisuutta ja toiminnallisuutta Silverlight-maailmaan ja toiset palvelut enemmän laajennettavuutta erilaisiin teknologioihin tukeutuen. Päällimmäisenä tarkoituksena on mahdollistaa olio-ohjelmointi sellaisille sovellustyypeille, joille ne eivät perinteisesti ole olleet mahdollisia.

Dataominaisuudet mahdollistavat LINQ:n käytön Silverlight-sovelluksissa sekä LINQ to XML -ominaisuuksia. Dataominaisuuksien avulla voidaan käyttää myös XML:ää tiedon varastointiin, ja se mahdollistaa sarjallistuvien (Serialization) luokkien käytön.

Perusluokkien kirjasto (BCL) tarjoaa kirjastoja .NET-ohjelmistokehyksestä, jotka mahdollistavat muun muassa merkkijonojen käsittelyä ja erilaisia kokoelmia.

Windows Communication Foundation (WCF) -mahdollistaa yhteydet etäpalveluihin ja tietoihin ja sisältää http-pyyntö ja vastausobjektin. Se tarjoaa myös tuen JSONin käyttöön, jolla mahdollistetaan kahden erillisen Silverlight-sovelluksen kommunikointi. WCF-palveluiden avulla voidaan käyttää myös LING to SQL -palveluita tiedon noutamiseen tietokantapalvelimilta.

Common Language Runtime (CLR) vastaa muistin hallinnasta, tyypin turvallisuuden tarkistamisesta ja poikkeuksien hallinnasta. Windows Presentation Foundation (WPF) tarjoaa Windowsista tuttuja kontrolleja käytettäväksi. Se sisältää napin, kalenterin, valintalaatikon, datataulukon, päivän valitsemisen, linkin, listakontrollin, radionapin sekä vieritysnäkymän.

Dynamic Language Runtime (DLR) tarjoaa mahdollisuuden käyttää erilaisia skriptikieliä Silverlight-sovellusten yhteydessä. Tällaisia kieliä ovat esimerkiksi JavaScript ja IronPython. DLR:n avulla voidaan myös lisätä tukea muille kielille, jotta niitä voitaisiin käyttää Silverlightin yhteydessä.

6 Silverlight-teknologian asettamat vaatimukset ohjelmistokehitykselle

Sovelluskehityksen aloittaminen Silverlight 3:lla asettaa vaatimuksia niin tarvittavien ohjelmistojen osalta kuin sovelluskehittäjän tietotaidollekin. Alkuun pääseminen on suhteellisen helppo prosessi, mutta Silverlight-maailma tarjoaa myös hyvin paljon syvyyttä. Erilaisia opeteltavia asioita ja tekniikoita on runsaasti.

6.1 Ohjelmistokehitysalusta

Ohjelmistojen vaatimusten osalta asiaa mietitään myös luvussa 9.1. Ensimmäinen vaatimus on ajan tasalla oleva ohjelmistokehitysalusta. Yksi vaihtoehto on lähteä liikkeelle asentamalla Visual Studio 2008 ja siihen tarjolla oleva Service Pack 1.

Tämän lisäksi tarvitaan Silverlight 3.0 Tools-paketti, jossa on tarvittavat kehitystuet Silverlight 3:a varten. Tämä lisää Visual Studioon muun muassa projektisapluunat (template), XAML-generaattorin, Silverlight-sovellusten testaus- ja vihreenkorjausominaisuuden, WCF-sapluunat sekä tuen Silverlight 3:n ulos selaimesta sovelluksille.

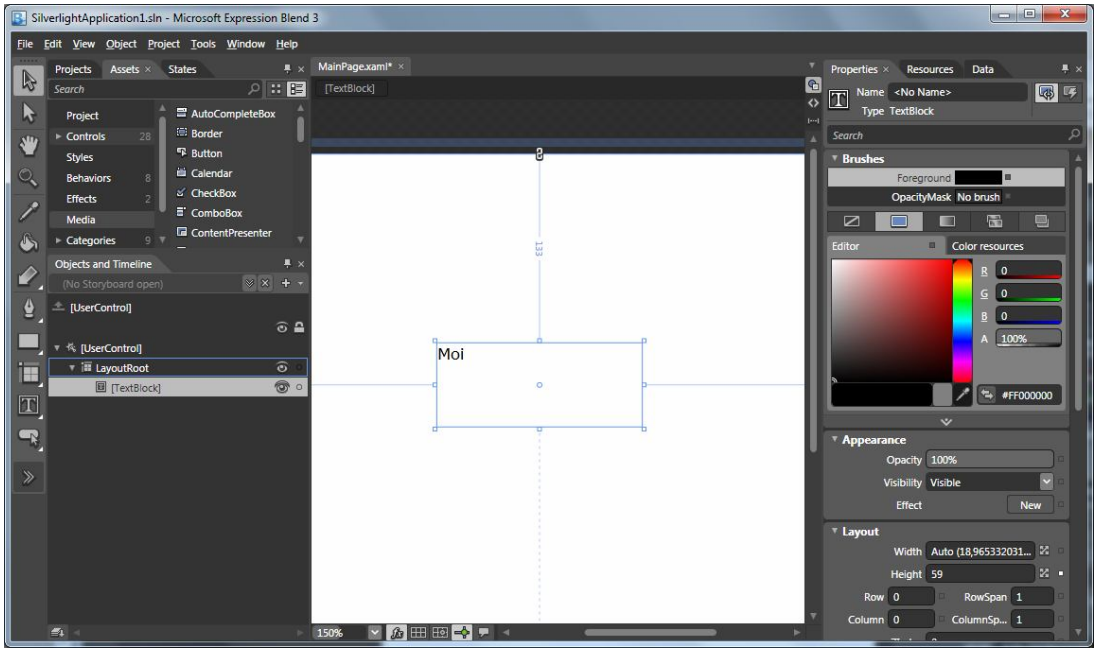
Vaihtoehtoisesti Silverlight 3 Tools voidaan asentaa myös Microsoft Visual Web Developer 2008 Expressiin, jossa on asennettuna Service Pack 1. Silverlight 3 Tools ei vaadi erillistä lisenssiä, vaan sitä voidaan kehittää Visual Studion lisenssillä.

Nämä ovat ohjelmistojen osalta minimivaatimukset ohjelmistokehitykselle, etenkin jos sovelluksen tulee kommunikoida esimerkiksi tietokantapalvelinten kanssa.

Visual Studiossa XAML:n toteuttaminen tehokkaasti on hankalaa. Varsinkin tehokasta editoria, josta lopputuloksen näkisi, ei ole tarjolla. Salkusta voidaan vetää komponentteja projektin XAML-sisällöksi ja Visual Studio generoi XAML-koodin. Tämä tarkoittaa ikävä kyllä ainoastaan komponentin aloittavaa ja päättävää tagia, esimerkiksi <TextBlock></TextBlock> tai <StackPanel></StackPanel>.

Visuaalisen toteuttamisen tehokkuuden puolesta vaatimuksiin voidaan lisätä myös hyvin suurella painotuksella Microsoft Expression Blend 3, joka tarjoaa alustan Silverlight 3-sovellusten kehittämiselle visuaalisella XAML-editorilla.

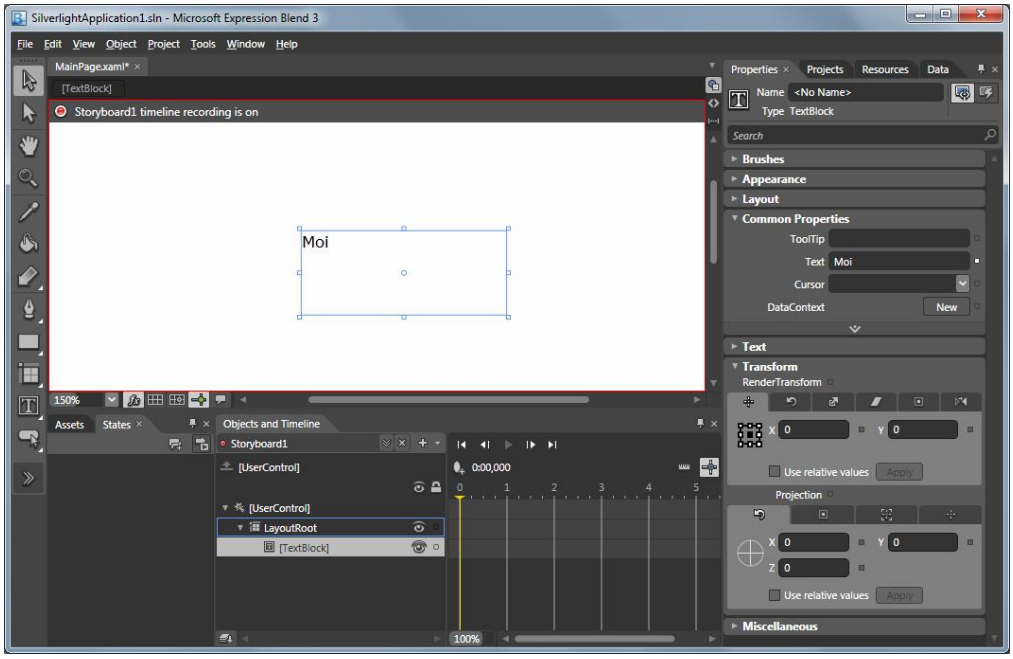
Blend on hyvin versiorajoitteinen. Vanhalla versiolla ei tietenkään voi kehittää Silverlight 3-ympäristöä, mutta Blend 3:lla ei voi myöskään kehittää Silverlight 2-sovellusta.



Kuva 7. Microsoft Expression Blend 3, jossa muokataan TextBox-elementtiä.

Blend on visuaalinen XAML-editori, joka mahdollistaa komponenttien vetämisen suoraan työkalusalkusta sovelluksen pohjana toimivalle elementille. Kuvassa 7 on nähtävissä Blend 3 toiminnassa. Blend generoi tämän jälkeen komponentin tiedoista ja asetetuista attribuuteista tarvitun XAML-koodin. Komponentteja voidaan sijoitella paremmalla tarkkuudella, eikä tarvitse arvuutella esimerkiksi pikseleiden määriä. Blendin käyttöliittymän avulla voidaan tarkastella komponenttien attribuuttien arvoja yhdellä silmäyksellä, myös niitä, joita ei ole asetettuna.

Blend tarjoaa myös erillisen tilan animaatioiden toteuttamiselle. Tila on esitetty kuvassa 8, mutta kuten komponenttienkin osalta, animaatiot luodaan sovelluksen XAML-koodin sekaan. Tämä ei ole aina se optimaalisin keino, mutta Blendin avulla on hyvin helppo tehdä erilaisia kokeiluja ja Blendin generoimasta XAML:stä pystyy hyvin nopeasti kirjoittamaan vastaavan C#-koodin ohjelman taustalle.



Kuva 8. Microsoft Expression Blend 3, storyboard edition tilassa.

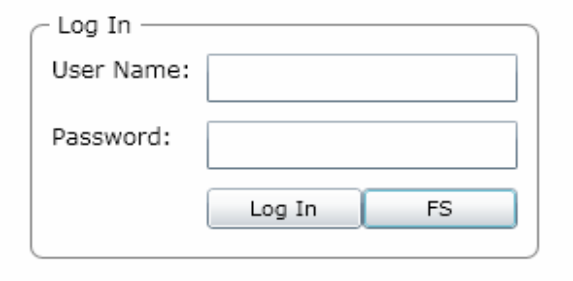
Blend 3 on osa Microsoft Expression Studio 3:a. Tätä työtä tehdessä XAML:n tuottamisen tehokkuus ja tarkkuus muuttui parempaan suuntaan Blendin avulla.

6.2 Ohjelmistokehittäjä

Ensimmäisenä vaatimuksena Silverlight-sovellusten kehittäjälle on XAML-tuntemus. Ilman XAML:a ei ole Silverlight-sovellusta. Tietysti koko sovellus voitaisiin toteuttaa C#:lla sovelluksen taustaosuudessa, mutta tämä ei ole aivan joka tilanteessa järkevää.

XAMLin oppimiskäyrää helpottaa suuresti XML:n osaaminen, koska XAML:n syntaksi on hyvin pitkälle samanlainen kuin XML:n. Kuvassa 9 näkyvän kirjautumisikkunan toteutuksen XAML on esitetty liitteessä 1.

XAML:n opettelun ohessa tulevat myös tutuksi Silverlightin komponenttien käyttömallit, eli mihin komponenttiin voidaan sijoittaa alakomponentteja ja mitkä komponentit ovat täysin itsenäisiä. Vaatimuksena voidaan näin kirjata ymmärrys Silverlightin peruskomponenteista.



Kuva 9. XAML-toteutus yksinkertaisesta sisäänkirjautumislomakkeesta.

Seuraavaksi vaatimuksiin voidaan lisätä .NET-ohjelmointikieli, jolla sovellukset saadaan ”elämään”. Vaihtoehtoina ovat C# tai VB.NET, joiden avulla viimeistään tulee tutuksi myös olio-ohjelmointi, joka esittäytyy merkittävässä roolissa tiedon sitomisen yhteydessä (luku 8.4) sekä käytettäessä MVVM-suunnittelumallia Silverlight-sovellusta toteutettaessa (luku 8.5).

Tietokannasta tiedon noutamista varten tulee ymmärtää, kuinka WCF-palvelut toimivat sekä kuinka LINQ to Sql:ää käytetään. Tietokantakyselyjä ei tehdä suoraan SQL:llä vaan LINQ:lla. WCF-palveluiden ohessa tulee huomioida erityisesti tiedon asynkronin noutaminen. Tästä kerrotaan lisää luvussa 8.4.

Tehokkaasti toimiessa on myös hyvä opetella tiedon sitominen. Vanhoissa internet-sovellustekniikoissa tiedot päivitettiin useasti paikkoihin yksitellen. Silverlight-sovelluksissa päivitetään luokkaa, jonka muuttujat on sidottu käyttöliittymässä esitettyihin komponentteihin, ja päivitetään tämän jälkeen varsinainen käyttöliittymä.

6.3 Vaatimukset suunniteltaessa Silverlight-sovellusta

Silverlight-sovellusten suunnittelemisessa ei ole kovin paljoa vaatimuksia. Vanhoissa tekniikoissa oltiin hyvin pitkälle riippuvaisia valmiista komponenteista ja elementeistä. Silverlight-teknologia menee tässä kohtaa ohi vauhdilla. Uusia komponentteja voidaan tehdä valmiiden Silverlight-komponenttien pohjalta muokkaamalla värimaailmaa tai vaihtoehtoisesti koko komponentin sapluunaa.

Suunnittelussa ei olla enää rajoittuneita siihen, mitä voidaan tehdä vaan pitkällä juoksulla vastaan tulee ajatus, mitä uutta voidaan keksiä. Uusien innovaatioiden ja komponenttien toteuttaminen on oma tarinansa.

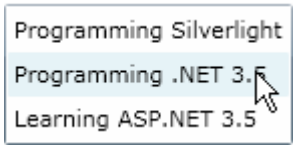
Komponenttien sijoittelun suhteen Silverlight-maailmassa voidaan työskennellä Grid-elementillä, joka vastaa toiminnallisuudeltaan ja idealtaan HTML:stä tuttua Table-elementtiä.

Vaihtoehtoisesti voidaan myös tarvittaessa käyttää Canvas-elementtiä, johon voidaan myös sijoittaa useampi komponentti. Canvas-elementin erikoisuutena on, että siihen lisätyt elementit määritellään paikalleen määrittelemällä matka Canvas-elementin vasemmasta reunasta ja Canvas-elementin yläreunasta.

Valmiita komponentteja voidaan muokata hyvin vapaasti, ja niiden ominaisuuksia voidaan myös laajentaa tarpeen mukaan. Käyttäjälle esitettyä tietoa voidaan sitoa erilaisiin komponentteihin, joiden avulla käyttäjä voi muokata tietoa.

Kuva 10. Kustomoitu listakontrollin sapluuna Silverlight-sovelluksessa.

Esimerkki komponenttien uudelleen muokattavuudesta voidaan nähdä kuvasta 10. Kyseessä on perinteinen listakontrolli (ListBox, HTML:stä tuttu Select – Option valinta), jonka mallinnetta on muutettu. Muokkaamaton listakontrolli on nähtävissä kuvassa 11.



Kuva 11. Tavallinen Silverlight ListBox-komponentti.

Muokatussa listakontrollin mallineessa esitetään yhdellä rivillä nyt kolme eri tietoa luokan muuttujista. Mallineeseen on koottu kirjan nimi, ISBN-numero sekä hinta. Sen sijaan että mallineessa esitettäisiin vain kirjan nimi. Mallinetta voitaisiin muokata vielä enemmän, lisäämällä esimerkiksi kuva jokaiselle elementille.

Kuten aiemmin on todettu, valmis Silverlight-sovellus pakataan XAP-tiedostoksi. Tiedosto pitää sisällään ohjelman lähdekoodit, resurssit, kaiken mikä siihen on Visual Studion puolella liitettynä. Käyttäjän navigoidessa sivulle, jossa on Silverlight-sovellus tai elementti, käyttäjälle lähetetään koko XAP-tiedosto. Ennen kuin koko XAP-tiedosto on siirtynyt, näkee käyttäjä koko tämän ajan vain prosenttiluvun latauksen edistymisestä.

Tämä asettaa omat vaatimuksen sovellukselle riippuen siitä, minkälaisten verkkoyhteyksien välityksellä sovellusta käytetään. Yrityksen sisäverkossa toimiessa viiveet ovat yleensä pieniä ja nopeudet suuria. Useamman megatavun paketti siirtyy näin nopeasti eikä käyttäjä tästä vaivaudu. Tilanne on toinen, jos sovellusta ajetaan mahdollisesti hitaampien internet-yhteyksien välityksellä.

Tämän seurauksena olisi hyvä pyrkiä minimoimaan XAP-tiedoston koko sijoittamalla esimerkiksi sovelluksen tarvitsemat suuremmat valokuvat muualle kuin varsinaisiin projektitiedostoihin. Näin ne tulevat ladatuksi vasta kun Silverlight-sovellus niitä tarvitsee eikä niitä sisällytetä varsinaiseen XAP-tiedostoon.

Luvussa 11 esiteltävää prototyyppiä toteutettaessa huomattiin, että Telerikin tarjoamat kirjastot kasvattivat XAP-tiedostoa melkoisesti. Varsinaisen ohjelman osuus tiedoston koosta jää hyvin pieneksi. On hyvä pitää mielessä, että käyttääkö seitsemää - kahdeksaa erilaista kirjastoa vai yrittääkö soveltaa ja pärjätä mahdollisimman pienellä määrällä kolmannen osapuolen kirjastoja.

Vaihtoehtoinen tapa XAP-tiedoston koon rajoittamiseeen on siirtyä modulaariseen Silverlight-sovellukseen. Tässä mallissa ohjelma koostuu pienemmistä osista, esimerkiksi jokainen uusi näkymänsä on oma kokonaisuutensa. Tämä pienentää ensimmäisenä ladattavan XAP-tiedoston kokoa, eikä näin ollen pidä käyttäjää pimennossa. Samalla Silverlight-projekti voidaan hajauttaa useammaksi pienemmäksi projektiksi.

Käyttäjän navigoidessa sovelluksessa paikkaan, joka on eriytetty omaksi kokonaisuudekseen, voidaan käyttäjälle informoida asiasta esimerkiksi latauspalkin avustuksella samalla, kun taustalla ladataan seuraava kokonaisuus, jota sovelluksessa tarvitaan. Näin ollen ohjelman käyttö on joustavampaa, eikä käyttäjän tarvitse odottaa iäisyyksiä sovellusten osien kanssa, joita hän ei mahdollisesti tarvitse.

Modulaarinen malli mahdollistaa myös Silverlight-projektin tehokkaamman kehityksen, kun useampi kehittäjä voi omalla tahollaan työskennellä oman kokonaisuuden parissa ja keskittyä tähän nimenomaiseen kokonaisuuteensa. Näin ei olla jatkuvasti riippuvaisia siitä, mitä toinen on saanut aikaan.

7 Silverlightin asettamia rajoituksia ohjelmistokehityksessä

7.1 Kuvaformaatit

Kuvaelementit tarjoavat täyden tuen JPEG-formaatille tai vaihtoehtoisesti PNG-formaatille (ei kuitenkaan 64-bittinen tai harmaasävy). GIF-formaatille ei ole tukea, koska on haluttu korostaa Silverlightin animaatiomallin ja perinteisen (ja yksinkertaisemman) animoidun GIF-kuvan eroja. Animoituja GIF-kuvia kun on käytetty perinteisessä internet-kehityksessä harmaasta alusta asti. (3, s. 147.)

Tämänkaltainen rajoitus ei tietenkään ole tyydyttänyt Silverlightin avulla ohjelmistoja tekeviä, vaan maailmalta on saatavissa erilaisia kolmannen osapuolen kirjastoja, joiden avulla voidaan saada tuki niin bmp- kuin esimerkiksi animoiduille gif-kuville.

7.2 Kokoruudun tilan rajoitukset

Sovellusta suunniteltaessa tulee huomioida ajoympäristö. Jos päädytään ajamaan sovellusta kokoruudun tilassa, se rajoittaa käyttäjän näkökulmasta ohjelman toiminnallisuutta. Ensimmäinen rajoittava tekijä on, että sovellusta ei voi pakottaa kokoruudun tilaan ohjelmallisesti.

Kokoruudun tilaan siirtyminen vaatii aina käyttäjän tekemän toimen (esimerkiksi napin painallus). Näin ollen kokoruudun tilaan siirtyminen on aina käyttäjän hallussa ja halussa. Tämän avulla Microsoft on halunnut suojella käyttäjiä joutumasta esimerkiksi käyttöjärjestelmäesittelyhuijauksen uhreiksi.

Näppäimistöltä ei kokoruudun tilassa hyväksytä kuin välilyönti, nuolinäppäimet, tabulaattori, home-, end-, page up-/down- ja enter-näppäin Tällä rajoituksella halutaan suojata käyttäjä mahdollisilta salasanojen spoofaukselta. Rajoituksen takia käyttäjän on mahdoton syöttää tekstiä tai numeroita syöttökenttiin kokoruudun tilassa. (4.)

Silverlight-liitännäinen ei myöskään tue tiedoston avaus- ja tallennusdialogeja, kun sovellusta ajetaan kokoruudun tilassa. Microsoft neuvoo vielä erikseen välttämään näiden dialogien käyttöä, jos ollaan kokoruudun tilassa, koska dialogin käyttö yrittää palauttaa sovelluksen ensin takaisin selaimen sisältä näytettäväksi. Tämä saattaa aiheuttaa toisissa selaimissa ristiriitaisia tapahtumia. Tarvittaessa voidaan ohjelmallisesti palata selaimen sisänäkymään ennen kuin avataan ko. dialogeja.

7.3 Fontit

Silverlightin kehitysympäristö sisältää kohtalaisen valikoiman fontteja, latinalaisen tekstin fontit on lueteltu liitteessä 2. Mikäli luettelosta ei löydy mieleistä ajaudutaan tilanteeseen, jossa hienon sovelluksen rakentamisen sijaan joudutaan ns. harmaalle alueelle.

Kuten aiemmin on todettu, valmis Silverlight-sovellus pakataan XAP-tiedostoksi. Myös tuntemattomat fontit laitetaan XAP-tiedostoon ihan sellaisenaan. Näin ollen kuka tahansa yllättävän pienellä vaivalla pystyy irrottamaan fontin sovelluksesta omaan käyttöönsä. Teknisesti katsoen tällaisessa tilanteessa Silverlight-sovelluksen luonut taho on fontin jakajana ja mahdollisesti pahassa välikädessä laillisesta näkökulmasta katsoen.

Kokonaisen fontin joutuminen XAP-tiedostoon voidaan kuitenkin kiertää mutta se vaatii kevyttä XAP-tiedoston muokkausta, sekä tämän lisäksi XAML:n tiedoston muokkausta tai CS-tiedostojen muokkausta, riippuen siitä, kummassa sovelluksen tekijä on määritellyt käytettävät fontit. Pahimmassa tapauksessa molempia tiedostoja saa muokata rankalla kädellä.

7.4 Tiedostojen lukeminen ja tallentaminen

Silverlight-sovelluksesta voidaan tallentaa tietoa käyttäjän koneelle ja lukea tätä tietoa, mutta tällaiseen ns. ”hiljaiseen toimintaan” on käytössä ainoastaan paikallinen tallennusvarasto.

Mikäli sovellus mahdollistaa, käyttäjä voi halutessaan tallentaa esimerkiksi kaavion kuvakaappauksen koneellensa, tämä onnistuu, erillisen lomakkeen kautta. Samoin on mahdollista rakentaa sovellus, johon käyttäjä esimerkiksi lähettää (upload) valokuvia. Tämäkin onnistuu, erillisen lomakkeen kautta.

Paikallinen tallennusvarasto on kooltaan selainsovelluksissa oletusarvoisesti yhden megatavun suuruinen, selaimen ulkopuolella toimivissa sovelluksessa koko on suurennettu 25 megatavuun. Oletuskokoa voidaan suurentaa, mutta Silverlight kysyy tähän suostumuksen käyttäjältä. Mikäli käyttäjä ei halua antaa lisätilaa sovellukselle, sovellus ei tätä tilaa saa.

Helpoin tapa katsoa sovelluksien varaamaa tilaa on mennä sivulle, jossa on jokin Silverlight-sovellus ja painaa hiiren kakkospainiketta sovelluksen päällä ja valita ponnahdusikkunasta avautuva Silverlight-valinta.

Tämän jälkeen välilehdeltä ”Application Storage” pystyy hallitsemaan sovelluksien varaamia tiloja tai vaihtoehtoisesti poistaa käytöstä koko sovellusten käyttämän varaston. Silverlight-sovellus voi itsenäisesti kirjoittaa ainoastaan tähän varastoon, eikä Silverlight-sovelluksella ole näin pääsyä käyttäjän tiedostojärjestelmään.

Tiedostojen luennassa tietoturvallisuuteen on myös kiinnitetty huomiota. Kuten kokoruudun tilaan siirtymisessä myös tiedoston lukemisen tarjoama lomake voidaan avata ainoastaan käyttäjän tekemän toimen jälkeen (esimerkiksi napin painallus).

8 Silverlight-teknologian käyttö

Silverlight-sovellus koostuu XAML:stä ja C#-koodista (tai VB.NET). XAML määrittelee sovelluksen käyttöliittymän ja C#-koodi taustalla vastaa ohjelman vuorovaikutuksesta. Esimerkiksi napin painalluksesta laukeaa tapahtuma, jossa on määritelty animaation aloitus. Animaatio on voitu toteuttaa XAML:n puolelle, mutta sitä voidaan kutsua napin ”click-eventissä”. Vaihtoehtoisesti animaatio voidaan myös luoda C#-koodilla.

Silverlight ei siis määrittele, tuleeko komponentit ja tapahtumat määritellä XAML:ssä vai sovelluksen taustalla olevassa koodissa. Jos käytössä on Expression Blend, sovelluksen ulkoasun pystyy piirtämään hyvin nopeasti. Blend tuottaa myös XAML:n sovellusta varten.

Expression Blend tarjoaa myös tehokkaan tavan animaatioiden tekemiseen erillisellä kuvakäsikirjoitus (storyboard) -editorillaan. Blend tekee käyttäjän liikkeistä XAML:n koodin, jossa animaation eri vaiheet ovat.

8.1 Silverlight-sovellusten lokalisointi

Lokalisoinnilla tarkoitetaan sovelluksen muokkaamista siten, että käyttöliittymä vastaa kaikilta osin paikallisen kulttuuriin tapoja. Tällä tarkoitetaan niin käytettävää kieltä, numerotietojen syöttöä sekä päiväyksien formaatteja, eli perusasioita käytännön elämästä, joita joudutaan miettimään monesti suunniteltaessa sovelluksia. Tämä on ajankohtaista etenkin sellaisten sovellusten kanssa, joita tullaan käyttämään maailmanlaajuisesti eri kulttuureissa.

Silverlight-sovelluksissa lokalisoinnista huolehtii CultureInfo-luokka, jonka avulla tiedon näyttäminen eri kulttuureille on hyvinkin helppoa. Luokan avulla voidaan pyytää järjestelmän CurrentCulture-objektia eli nykyistä kulttuuria tai CurrentUICulture-objektia, joka sisältää nykyisen käyttöliittymän kulttuuriobjektin.

Asetusta voidaan myös vaihtaa halutuksi. Näin ollen voidaan tarjota käyttäjälle mahdollisuus valita käytettävä kieli, samoin käytössä oleva kulttuuriasetus päivämäärien ja numeroiden osalta. Tämä mahdollistaa ohjelman käyttämisen eri maissa sijaitsevilta päätelaitteilta, kuitenkin sellaisena kuin käyttäjä haluaa.

```
…
// Koodi tulostaa nykyiset kulttuuri informaatiot
outputBlock.Text += String.Format("The current culture is '{0}'.\n",
CultureInfo.CurrentCulture.Name);
outputBlock.Text += String.Format("The currrent UI culture is
'{0}'.\n", CultureInfo.CurrentUICulture.Name);
//       The current culture is 'fi-FI'.
//       The current UI culture is 'fi-FI'.

//kulttuurin vaihtaminen amerikan malliin
CultureInfo usaCulture = new CultureInfo("en-US");
Thread.CurrentThread.CurrentCulture = usaCulture;
Thread.CurrentThread.CurrentUICulture = usaCulture;

…
```

Kulttuuri muodostuu aina pienellä kirjoitetusta kielikoodista ja suurella kirjoitetusta maakoodin yhdistelmästä. Kuten edellä olevasta esimerkistä voidaan havaita, suomen tapauksessa kielikoodina on fi ja maakoodina on FI. Yhdysvaltain malli on hieman havainnollisempi, kielikoodina on englanti (en) ja varsinaisena maakoodina US.

Lokalisoinnin ansiosta käyttäjää ei tarvitse pakottaa syöttämään päivämääriä vieraassa formaatissa (suomen dd-mm-yyyy vs. amerikan malli mm/dd/yyyy), eikä tarvitse tehdä kompromisseja. Samoin sovelluksista tulee viimeistellympiä, kun käyttäjä näkee haluamansa valuuttaa oikein muotoiltuna (tuhat- ja desimaalierottimet, valuuttamerkin sijainti).

8.2 Kahden Silverlight-elementin kommunikointi keskenään

Kahden Silverlight-kokonaisuuden kommunikointi keskenään on havainnollistettu kuvassa 5 (ks. s. 19), myös kahden erillisen Silverlight-sovelluksen keskenään kommunikointi onnistuu. Näin voitaisiin toteuttaa esimerkiksi sovellus, jonka pääikkuna on yhdellä selaimen välilehdellä ja siitä ladattavat seuraavat ohjelman vaiheet selaimen muilla välilehdillä ja sovellukset pystyisivät kommunikoimaan keskenään tästä huolimatta.

Toiminnan pohjana on Silverlightin tarjoama LocalConnection API, joka pohjautuu Jsoniin (JavaScript Object Notation). Kahden erillisen sovelluksen välillä kommunikoi jokin luokka. Tämä luokka tulee toteuttaa niin, että se on serialisoituva (serialization) eli luokka osaa kirjoittaa itse tietonsa muistivirtaan ja tämän lisäksi lukea tietonsa virrasta (tämä edellyttää luokan metodien Serialize ja Deserialize toteuttamista), koska Jsonin avulla lähetettävät ja vastaanotettavat viestit ovat merkkijonoja.

Normaalissa .NET ajoympäristössä on tarjolla erilaisia sarjallistumiskykyjä. Näitä ei ole kuitenkaan tuotu Silverlight 3:n maailmaan, joten luokan kanssa joutuu tekemään hieman enemmän töitä.

Sovelluksen taustakoodiin tulee lisätä LocalMessageReceiver ja -Sender. Nimiensä mukaisesti LocalMessageSender lähettää sarjallistuneita luokkia eteenpäin ja LocalMessageReceiver vastaanottaa merkkijonovirtaa. Tämän lisäksi tulee huolehtia, että luokka sarjallistuu tarvittaessa ja se lähetetään eteenpäin.

Esimerkiksi kun kuvassa 5 näkyvässä sovelluksessa valitaan ylemmästä luettelosta joku henkilö, luettelossa tapahtuu SelectionChanged-tapahtuma. Tapahtumassa voidaan yksilöidä henkilö luettelon käyttämästä kokoelmasta. Henkilöluokka tulee sarjallistaa, ja tämän jälkeen se voidaan lähettää eteenpäin. Tiedot lähetetään eteenpäin asynkronisesti.

LocalMessageReceiverille tulee luoda tapahtumankäsittelijä MessageReceived-tapahtumalle. Tämän tapahtumankäsittelijän vastaanotetuissa tapahtuma argumenteissa on toisen sovelluksen lähettämä merkkijono. Merkkijonosta rekonstruoidaan henkilö. Tämän jälkeen henkilön tiedot täydennetään toiseen sovellukseen, kuten kuvasta 5 voidaan nähdä, esimerkissä vastaanottajana on taulukkomuotoinen sovellus.

Henkilöluokan konstruoinnin jälkeen tiedot päivittyvät lomakkeelle, jonka kenttiin henkilöluokan muuttujat on sidottu. Näin ollen ohjelman kehittäjä huolehtii vain luokan tietojen päivittämisestä, ja tietojen päivittämisen jälkeen Silverlight huolehtii uuden tiedon hahmontamisesta lomakkeelle.

8.3 WCF Service – LINQ-datan noutaminen tietokannasta

Internet-sovellusten tukena on lähes aina jonkinlainen tietokanta, johon sovelluksesta kommunikoidaan. Silverlight-maailmassa tietokannasta noudetaan tietoa seuraavan mallin mukaan.

Ensin lyhyesti kerrottuna tietokantaan menevä kysely kirjoitetaan LINQ:lla. Tämän jälkeen käytetään WCF Web-palvelua tiedon välittämiseen palvelimelle ja tämän jälkeen tiedon välittämiseksi palvelimelta takaisin Silverlight-sovellukseen.

Asiaan perehtyminen aloitetaan projektin luonnista, projekti tulee olla ”ASP.NET Web Application Project”. Tällä projektipohjalla aloitettaessa Visual Studio muodostaa Silverlight-osuuden, joka on nimettynä syötetyn projektin nimellä. Tämän lisäksi projektipuusta löytyy Projektinnimi.Web-osuus. Osuus sisältää sivun, jolla Silverlight-projekti esitetään, ja tähän osuuteen tullaan lisäämään LINQ:n kyselyt sekä WCF-palvelut, joiden avulla kommunikoidaan.

LINQ:ta varten Web-projektiin tulee ensin lisätä ”LINQ to SQL Classes”-luokka. Tämän jälkeen projektiin tulevaan DBML-tiedostoon voidaan liittää tauluja ja näkymiä tietokannoista, joihin Visual Studiosta on yhteys (Server Explorer). LINQ-luokka kannattaa lisätä ennen WCF palvelua, näin voi olla helpompi kirjoittaa WCF-palvelua, kun Visual Studio tunnistaa haettavat objektit LINQ:n käyttämän DBML-tiedoston pohjalta. Tämä edellyttää, että sovelluksessa käytettävät tietokannantaulut on liitetty kyseiseen tiedostoon.

Seuraavaksi projektiin tulee lisätä WCF Service (esimerkissä lisättävän palvelun nimi on Palvelu1). Tämän jälkeen projektista löytyy Palvelu1.cs, johon muodostetaan rajapinnat, joita kutsutaan Silverlight-projektista. Rajapintojen ollessa valmiita tulee niitä vastaavat LINQ-kyselyt kirjoittaa Palvelu1.svc.cs-tiedostoon.

Seuraavaksi liitetään WCF-palvelu Silverlight-projektiin muodostamalla referenssi palvelusta. Referenssin muodostamisen jälkeen se näkyy Silverlight-projektin alla kansiossa Service References. Palveluita muokattaessa tai niitä lisättäessä tulee palveluiden referenssi päivittää. (16.)

Parhaan varmistuksen ohjelmistokehittäjä saa asiasta, kun napsauttaa hiiren kakkospainikkeella palveluiden referenssitiedostoa ja käskee Visual Studiota päivittämään referenssin. Ainakaan perusasetuksilla tätä ei tapahdu automaattisesti projektia käännettäessä, mikä saattaa aiheuttaa sekaannusta, kun korjattu LINQ-kysely palauttaakin saman tuloksen kuin ennen korjausta.

Palveluiden referenssin ollessa kunnossa ne voidaan sijoittaa Silverlight-projektiin. Käytettäessä palveluita tulee ensin luoda nk. webService. Tähän liitetään tapahtumankäsittelijä, joka yleisesti nimetään ”Palvelu1Tapahtunut”. Tapahtumankäsittelijässä hoidetaan palvelun palauttaman tiedon jatkokäsittely.

Tapahtumakäsittelijä nimetään menneeseen aikamuotoon siitä syystä että palvelut ovat asynkronisia. Ne eivät siis tapahdu samalla silmänräpäyksellä kuin nappia painetaan, vaan sitten kun Silverlight-sovellus katsoo parhaaksi. Palvelut eivät myöskään estä ohjelman suoritusta, eli ei voida vain jäädä odottamaan palvelimen vastausta ja turvautua ohjelmassa, että seuraavassa vaiheessa tiedot ovat tulleet palvelimelta.

8.4 Tiedon sitominen

Tiedon sitominen (data binding) vastaa pitkälle .NET kehyksen vastaavaa toimintoa, jossa tietoa sidotaan olemassa oleviin elementeihin. Tämä on yksi Silverlightin tehokkaimmista tekniikoista, jolla tietoa tuodaan sulavasti käyttöliittymään. Esimerkiksi WCF Servicen avulla tuotetaan listamainen tieto jostain tietokannan taulusta. Tämä lista sidotaan käyttöliittymässä olevaan Data-tauluun, näin ollen tietokannan taulu tietoineen on tuotu käyttöliittymään ja käyttäjä voi muokata taulussa näkyviä tietoja. Tiedon tallentamisesta huolehtiminen on oma tarinansa.

Tiedon sidonnassa voidaan lisäksi määrittää, onko sidottu data yksi- vai kaksisuuntaista, kaksisuuntaisessa datan muutokset tallennetaan olioon. Yksisuuntaisessa tietoa vain luetaan oliosta, mutta kaksisuuntaisella tiedon sitomisella tietoa voidaan suoraan muokata sitä esitettävässä elementissä (jos elementti tukee tätä) ja tieto tallentuu olioon.

Esimerkiksi oliosta on sidottu kokonaisluku sovelluksessa näkyvään liukupalkkiin ja liukupalkin vieressä on esitetty arvo. Liukupalkkia siirtämällä luvun arvo muuttuu siinä missä liukupalkin arvo muuttuu. Tiedon muutos ei automaattisesti tallennu muualle kuin olioon. Oliosta tuleekin tehdä näin ollen sellainen, että tämänkaltaiset muutokset tallennetaan.

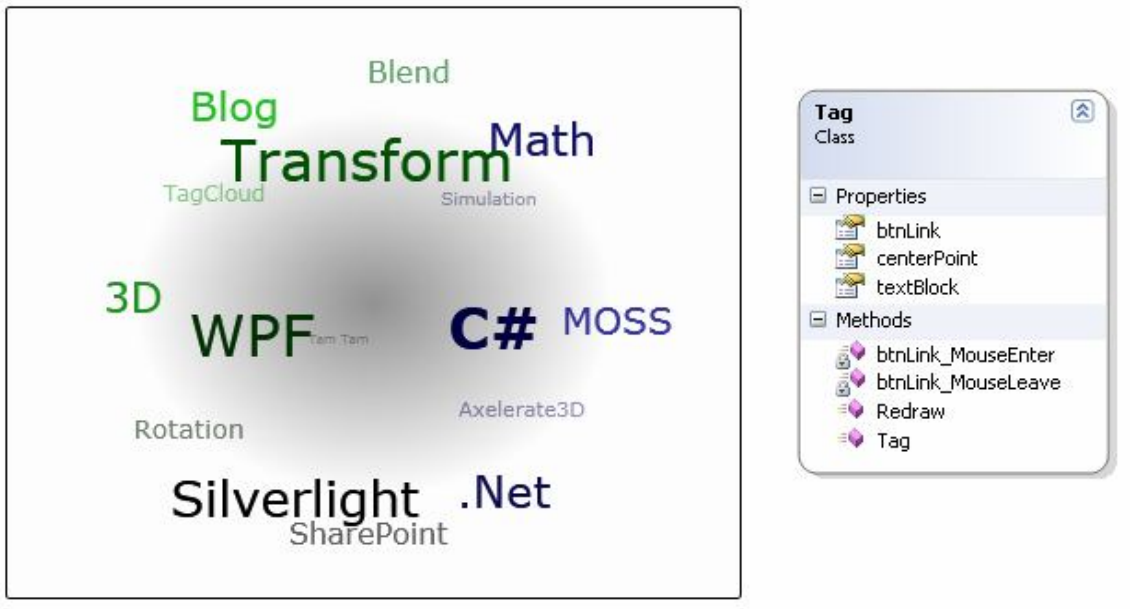
Tiedon sitominen on todella tehokasta. Kaikki käyttöliittymässä tarvittava tieto voidaan sitoa esimerkiksi kokonaan yhteen luokkaan ja tätä luokkaa päivitetään muista luokista. Näin ollen kaikki toiminta tapahtuu sovelluksen Code-Behindissa, kun luokat työskentelevät muuttuvan datan kanssa tai noutavat pyynnöstä uutta dataa näytettäväksi, esimerkiksi kuvassa 12 pilven oliot ovat kaikki yksittäisiä Tag-olioita, joiden tiedot on sidottu pilvessä esiintyviin elementteihin.

Esimerkiksi sovelluksen XAML-puolella sijoitetaan sovellukseen yksi DataGrid:

```
…
<my:DataGrid x:Name="theDataGrid" AlternatingRowBackground="Beige"
AutoGenerateColumns="True" Width="700" Height="500"
CanUserResizeColumns="True" />
…
```

Sovelluksen Code-Behindissa voidaan luoda esimerkiksi lista, joka tämän jälkeen sidotaan DataGridiin. Näin ollen saadaan nopeasti elementti, jossa voidaan helposti esittää tietoa tietokannan taulusta.

DataGridin ominaisuuksista tulee huomioida vierityksen onnistuminen sekä sarakkeiden mukaan tiedon järjestäminen (nousevasti tai laskevasti). Kuten edellä olevasta koodista voidaan havaita, kontrollin parametreilla voidaan hallita suuresti kontrollin ominaisuuksia.



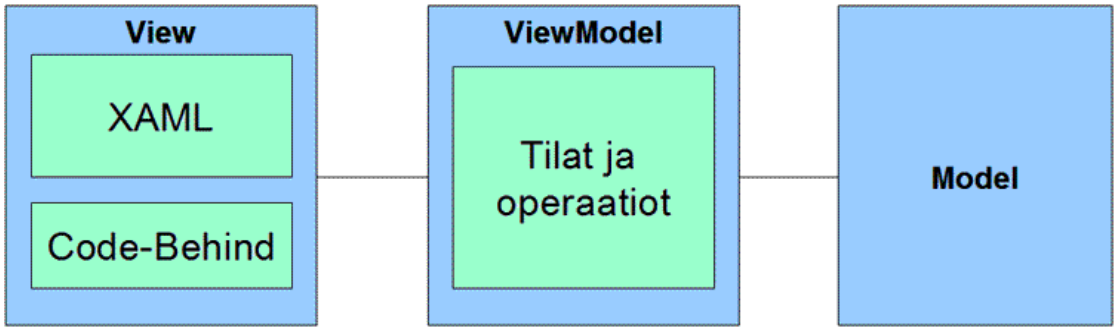
Kuva 12. Esimerkki tiedon sitomisesta, pilvessä näkyvät elementit muodostuvat Tag-luokan kokoelmasta. Jokainen elementti on tietoinen omasta sijainnistaan (centerPoint), sekä omasta tekstistä (joka on sijoitettu textBlock muuttujaan).

8.5 Model-View-ViewModel -arkkitehtuurimalli

Vuonna 2005 John Gossman (Microsoftin WPF- ja Silverlight-arkkitehti) esitteli ensimmäisen kerran MVVM-mallin. Kyseinen arkkitehtuurimalli on hyvin lähellä monista muista ohjelmointikielistä tuttua MVC-mallia (Model-View-Controller) ja tästä syystä malleilla on kaksi hyvin isoa yhtäläisyyttä.

Kuten MVC-mallissa, myös MVVM-mallissa malli (Model) on vastuussa kaikesta logiikasta ja tiedon noutamisesta ja tallentamisesta. Näkymä (View) on vastuussa käyttöliittymässä näkyvien elementtien piirtämisestä kuten MVC:n vastaava näkymä, tämä sisältää varsinaisen XAML:n, jota Silverlight tulkitsee.

MVVM-mallia suositaan Silverlight-sovellusten kehittämisessä, koska Silverlight on nimenomaan tehokas tiedon esittäjä ja varsinainen sovelluslogiikka halutaan sijoittaa käyttöliittymän taustalle. Tämä on jälleen yksi ominaisuus, jolla mahdollistetaan sovelluksen kehittäminen monen kehittäjän toimesta yhtä aikaa.



Kuva 13. MWWM-arkkitehtuurimalli

Yksinkertaisimmissa Silverlight-esimerkeissä tätä mallia on yleensä sivuutettu. Esimerkeissä on ollut jokin käyttöliittymä, jonka Code-Behindissa tiedot ladataan palvelimelta tai tiedostosta ja samalla käyttöliittymä on ollut vastuussa muun muassa näkymän logiikasta ja mahdollisista käyttäjän syötön tarkistuksista.

Kun kaikki sovelluksen koodi on samassa paikassa, on hyvin suuri mahdollisuus sotkeutua siinä, mikä olikaan varsinaisen esityksen tarpeeseen ja mikä oikeata logiikkaa. Tällaisella tavalla tehtyyn sovellukseen on myös hyvin hankala palata jälkikäteen, vaikka tämä on kieltämättä se nopein tapa tiedon esittämiseen, mutta soveltuu lähinnä nopeiden prototyyppien tarpeeseen.

Kuten kuvasta 13 nähdään, ViewModel sijoittuu näkymän ja mallin väliin. Kuten nimikin antaa viitteitä, kyseessä on näkymän malli, joka pitää sisällään mallilta saadun datan siinä muodossa kuin sitä tarvitaan mallissa.

Esimerkiksi mallissa on jokin liukuluku, jota halutaan esittää käyttäjälle, esimerkiksi mittarikontrollin avulla. Mittariin voidaan sitoa suoraan tämä liukuluku, mutta käyttäjää varten on selvyyden vuoksi parempi muuntaa luku pienemmälle tarkkuudelle (vähemmän desimaaleja) ja esittää tämä tieto.

Lisäksi ViewModeliin sijoitetaan käyttöliittymän taustalogiikkaa ja operaatiot luodaan metodeina, joita näkymä kutsuu. Asian ydin on, että ViewModel ei ole riippuvainen näkymän ratkaisusta.

Asiantuntijat neuvovat että näkymästä tulisi päästä eroon mahdollisimman pitkälle Code-Behind-koodista, toiset sallivat siellä olevan vähän näkymän tarvitsevia toimintoja esimerkiksi sellaisia, joilla ei ole mitään tekemistä mallin tai ViewModelin kanssa.

Lyhyesti kuvattuna ViewModeliin noudetaan mallista data ja muunnetaan se sellaiseen muotoon, että se voidaan helposti sitoa näkymään. (10.)

8.6 Silverlight-sovelluksen optimointia komponenttien näkyvyydellä

Silverlight-sovelluksia tehdessä, etenkin sovelluksia, jotka koostuvat useammasta animaatiosta ja kuvakäsikirjoitustaulusta (storyboard), voi helposti tulla vastaan tilanne, jossa tietokoneen prosessorin tehoraja tulee vastaan ja animaatiot alkavat hidastua ja pätkiä, etenkin vanhemmilla laitteilla.

Yksi tapa optimoida Silverlight-sovellusta on varmistaa että objektit, joita käyttäjä ei pysty havaitsemaan, ei myöskään hahmonneta. Silverlight kun hahmontaa objektit aina, kun ne ovat aktiivisena kuvakäsikirjoitustaulussa, on niiden päällä sitten toisia objekteja tai ei.

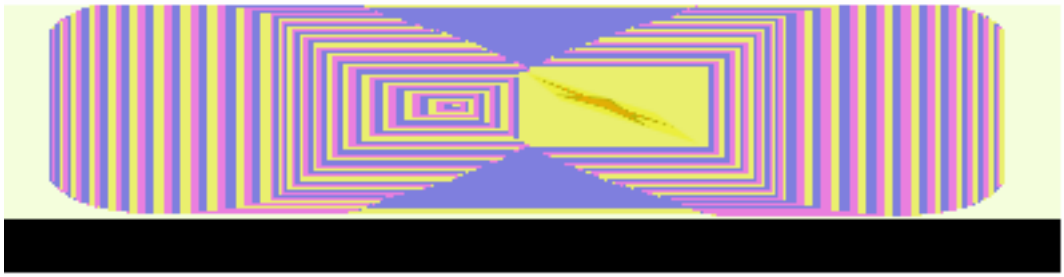
Kuva 14. Yksinkertainen animaatio Silverlightilla.

Kuvassa 14 on yksinkertainen animaatiototeutus Silverlightin avulla. Animaatiossa objekti liikkuu varatussa tilassa vasemmalta oikealle jatkuvasti ja samalla tähteä käännetään 180-astetta x-akselinsa ympäri.

Muuttamalla tähti objektin Visibility-parametri muotoon ”Collapsed” tähti katoaa ruudulta mutta tausta jää. Tämä on oikea tapa käsitellä objekteja, kun niitä ei väliaikaisesti enää tarvita. Tämä kertoo myös Silverlightille, että objektin tila ei kiinnosta käyttäjää ja se voidaan jättää hahmontamatta.

Toinen vaihtoehto olisi peittää tarvittaessa tähtiobjekti esimerkiksi Canvas-elementillä. Vaikka käyttäjä ei tähti objektia tämän jälkeen näe, se hahmonnetaan silti jok’ikisellä kehyksellä (frame). Monen objektin kanssa tästä aiheutuu todella iso kuorma prosessorille, mistä seuraa edellä kuvattuja ongelmia.

Ohjelmistokehittäjä, jolla on pääsy HTML-dokumenttiin, josta Silverlight-sovellus käynnistetään, pystyy kytkemään toimintaan ominaisuuden, joka paljastaa hyvin äkkiä, mitä elementtejä hahmonnetaan missäkin vaiheessa ja mistä voitaisiin mahdollisesti saada vapautettua jo mahdollisesti tuhlattuja resursseja.



Kuva 15. Silverlight-sovellus jossa enable redraw regions toiminnassa.

HTML-sivulle lisättävä muuttuja on nimeltään enable redraw regions, eli näytetään uudestaan piirretyt alueet. Esimerkki lisätystä muuttujasta HTML-koodissa:

```
…
<param name=“enableRedrawRegions” value=“true” />
…
```

Tämän muutoksen jälkeen, kun Silverlight-sovellus käynnistetään uudestaan, voidaan havaita kuvassa 15 näkyvää väritystä. Jokainen pikseli, joka piirretään uudestaan, värjäytyy. Näin ollen voidaan hahmottaa tähti objektin liike. Vaikka se peitetään Canvas-elementillä, sen piirtäminen jatkuu yhä, ennen kuin sen Visibility-parametri muutetaan arvoon Collapsed.

9 Silverlight-teknologian käyttöönotto

9.1 Kehitysympäristössä

Silverlightin kehityksen aloittamiselle on olemassa kaksi hyvin eri polkua. Voidaan aloittaa enemmän tekniseltä kannalta tai vaihtoehtoisesti enemmän visuaaliseen näkökulmaan pohjautuvasta työskentelystä.

Tässä insinöörityössä käytetty Silverlight 3 vaatii tekniseltä näkökulmalta Visual Studio 2008:n. Työ voidaan myös aloittaa visuaalisella suunnittelulla Microsoft Expression Blend 3:lla. Molemmat työkalut mahdollistavat Silverlightin käyttämän XAML:n kirjoittamisen. Visual Studiossa on myös olemassa yksinkertainen editori XAML-koodin tuottamasta sivusta. Microsoftin dokumentaatio kertoo, että kyseistä editoria ei olisi Silverlight 3:n mukana alkuunkaan. Asiaa on perusteltu editorin vaatiman ylimääräisen tehon tarpeella ja siinä ilmenneillä hahmontointi ongelmilla.

Editori on ensimmäisen version aikana ollut vastaavanlainen kuin muut Visual Studiossa olevat tutut editorit, eli komponentteja on voinut siirrellä lomakkeelle (tässä tapauksessa sivulle) ja tämän jälkeen vielä liikutella haluamaansa paikkaan. Silverlight 2:n myötä editori muuttui ”vain luku”-tilaan. Komponentit vedettiin työkaluista XAML:n sekaan, ja editori näytti, miltä sivu näyttää. Kolmosversiossa editori on piilotettu, mutta se löytyy vielä, tosin edelleen yhtä yksinkertaisena kuin kakkosversiossa.

Visual Studio tarjoaa paremman alustan ja työkalut UserControllin taakse kirjoitettavan koodin tekemiseen, kun taas Blend tarjoaa paljon enemmän visuaalisen puolen tekemiseen. Lisäksi Visual Studio on täysin omassa kategoriassaan, kun tulee tarve toteuttaa sovelluksia, joissa esitetään tietoa tietokannasta.

Kolmansien osapuolten tarjoamat lisäkirjastot toimivat Visual Studion puolella moitteetta, mutta Blendin kanssa saattaa esiintyä pieniä ongelmia. Esimerkiksi kaikki kolmannen osapuolen kirjaston tarjoamat komponentit eivät näy täysin oikein Blendissä. Tämä ei vaikuta komponenttien toimintaan itse sovelluksessa kuitenkaan.

Ensimmäistä kertaa, kun asennettiin Silverlight 3.0 Toolseja jo olemassa olevan kehitysympäristön (Visual Studio 2008 SP1) tueksi, ilmeni ongelmia. Asennus eteni hetken matkaa, kunnes ilmoitti ”*Original exit code: <path>\Silverlight 3.0 Tools\Silverlight_Developer.exe returned non-MSI error code: 0x5e2 - (null)*”.

Ensimmäinen ajatus oli heti, että ongelmaan löytyy vastaus Silverlightin yhteisöstä (www.silverlight.net). Silverlightin sivuston auetessa kone ilmoitti ensimmäisenä, että koneessa ei ole tukea Silverlightille. Tämä aiheutti epäilyksen, että varsinaisten työkalujen asennus olisi poistanut Silverlightin tuen, koska se oli olemassa ennen kuin asennusta aloitettiin. Tämä aiheutti hieman hämminkiä, josta etenemissuunnaksi valittiin koneen uudelleenkäynnistys, jonka jälkeen otettaisiin uusi yritys Silverlight 3.0 Tools-paketin kanssa.

Uudelleen käynnistämisen jälkeen testattiin vielä, miten oli Silverlight-tuen laita. Tukea ei ollut, tämä vahvisti epäilykset, että asennuspaketti oli enempää ilmoittelematta poistanut olemassa olleen tuen. Tämän jälkeen kokeiltiin ajaa asennuspaketti uudestaan. Tällä kertaa asennus soljui läpi normaalisti ja ilmoitti lopuksi kaiken menneen, kuten pitikin.

Mielenkiinnosta heti asennuksen jälkeen tuli testattua, mikä oli selaimen Silverlight-tuki nyt. Testi tuli suoritettua samalla sivustolla vierailemalla kuin edellisellä kerralla. Sivuston avautuessa myös sivuston Silverlight-komponentti latautui, ja näin voitiin todeta asennuspaketin palauttaneen tuen Silverlight-komponenteille.

Insinöörityötä tehdessä kehitysalustana toiminut tietokone vaihtui uudempaan. Uudemmalla koneella kehitysympäristöä asennettaessa ei vastaavaan ongelmaan törmätty. Silverlight Tools asentui puhtaasti ilman virheilmoituksia. Käyttöjärjestelmä oli molemmissa koneissa täysin sama (Windows XP, jossa SP3). Lisäksi käytettävä Visual Studion versio oli sama. Niin ikään Visual Studioon oli asennettu tarvittavat päivitykset (Service Pack 1).

9.2 Palvelimella

Silverlightin palvelinympäristön ei tarvitse olla Microsoftin IIS niin kuin voisi ensi alkuun kuvitella. Silverlight toimii useimmilla saatavilla olevilla palvelimilla (muun muassa Apache, IIS, LightTPD, Sun Java System Web Server). Esimerkiksi Apache pyöritää Silverlight-materiaalia ihan siinä missä IISkin. Tämä edellyttää, että palvelimen MIME-tyypit on konfiguroitu oikein. MIME-tyypeistä tulee löytyä Silverlightia varten .xaml- ja .xap-tyypit. Tarkemmat tiedot on esitetty taulukossa 1.

Taulukko 1. Silverlightin vaatimat MIME-tyypit

Tiedoston pääte	MIME tyyppi
.xaml	application/xaml+xml
.xap	application/x-silverlight-app

Microsoft on huomioinut tämän tarpeen omassa Server 2008:ssa (IIS 7.0) alkaen. Sen sijaan IIS 6.0:een nämä tiedot tulee lisätä. Palvelimen tunnistaessa MIME-tyypin oikein on käyttäjän selaimesta riippuvainen, toimiiko Silverlight sisältö vai ei.

Tilanteessa, jossa ei pystytä muokkaamaan palvelimen MIME-tyyppejä, ei välttämättä olla vielä umpikujassa. Microsoftilla Silverlight-kehitystyötä tekevä Tim Sneath (5) paljastaa blogissaan seuraavan tekniikan ongelman kiertämiseksi.

Uudellennimetään XAP-tiedosto ZIP-tiedostoksi, tämän jälkeen muokataan HTML-sivua, jossa Silverlight-elementti ladataan vastaamaan .XAP -> .ZIP-nimenmuutosta. Tämä on toisaalta sen verran radikaali toimenpide, että tätä voidaan suositella vain Silverlightin ensimmäisiä sovellusharjoituksia tehtäessä.

9.3 Käyttäjän näkökulmasta

Käyttäjän näkökulmasta Silverlightin käyttöönotto ei ole kovinkaan monimutkainen prosessi, eikä se eroa ollenkaan muiden vastaavien lisäosien asennuksesta.

Kuva 16. Silverlight-sovellus ilmoittaa, että asentamalla Silverlightin elämys kasvaa

Käyttäjän navigoidessa sivulle, jossa on jokin Silverlight-elementti, ja käyttäjälle ei ole käytössä olevassa selaimessa Silverlight-tukea, elementin paikalle tulee vaihtoehtoinen materiaali, jossa yleensä kehoitetaan asentamaan Silverlight. Esimerkki tällaisesta ilmoituksesta on esitetty kuvissa 16 ja 17.

Kuva 17. Silverlight-sovellus ilmoittaa, että asentamalla Silverlightin elämys kasvaa.

Ohjelmoija pystyy vaikuttamaan, mitä käyttäjälle näytetään tilanteessa, jossa käyttäjän selaimessa ei ole Silverlight-tukea. Kuvassa 17 voi nähdä tämänkaltaisen ilmoituksen sivustolta www.silverlight.net ja kuvassa 16 on perinteisempi ilmoitus aiheesta.

Vaihtoehtoinen materiaali on ohjelmoijan kannalta helppo määritellä. Edellä olevassa esimerkissä vaihtoehtoinen sisältö on HTML-sivulla, josta Silverlight-sovellus aloitetaan. Varsinaisen Silverlight-sisällön esittävien <object>-tagien sisälle sijoitetaan vaihtoehtoinen sisältö määriteltyjen parametrien jälkeen:

```
…
<object>
    <param/>
    <param/>
    <img src=”vaihtoehtoinen kuva”/>
</object>
…
```

Linkkiä tai kuvaa napsautettuaan käyttäjä lataa koneelleen Silverlight.exe (Windows) -tiedoston, joka sisältää asennusohjelman ja itse Silverlight-lisäosan. Mozillan Firefoxiin kyseisen lisäosan asennustiedosto on kokoa 4,7 megatavua (kyseessä Silverligt 3:n asennuspaketti, tarkennettuna versio 3.0.50106). Asennuspaketin avaamisen jälkeen Silverlight-liitännäinen opastaa asennuksessa. Asennusprosessin aloitus on esitetty kuvassa 18.

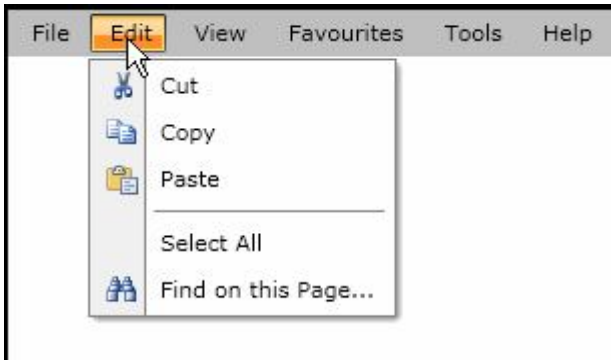
Kuva 18. Silverlight-lisäosan asennuksen aloitusruutu.

Asennuksen jälkeen käyttäjän tulee päivittää sivu, josta Silverlight-lisäosan asennus aloitettiin. Kaiken ollessa kohdallaan Silverlight-sovellus näyttää sisältönsä.

Silverlightin tukemat käyttöjärjestelmä-selainkokoonpanojen matriisi on esitetty liitteessä 4.

10 Silverlight-teknologian tarjoama lisäarvo web-sovellukselle

Silverlight-teknologian eduista ensimmäisenä asiana mainittakoon sovelluksen kokonaisuus. Silverlight-teknologian avulla voidaan toteuttaa web-sovellukset kuten työpöytäsovellukset. Näin niistä saadaan laadukkaamman oloisia kokonaisuuden ollessa yhtenäinen. Kuvassa 19 voi nähdä yksinkertaisen valikkototeutuksen Silverlight-sovelluksessa.



Kuva 19. Valikkoratkaisu Silverlight-sovelluksessa.

Edeltävillä tekniikoilla monesti tuli eteen tilanne, jossa joudutaan päivittämään koko sovelluksen sivu tai suurin osa sivusta. Tämä aiheuttaa ikävää elementtien välkkymistä ja lataustaukoja. Silverlightin avulla voidaan tuottaa uusi sisältö käyttäjän näkymättömiin ja tuoda tämä vanhan sisällön paikalle.

Silverlight-maailmassa voidaan helposti toteuttaa myös erilaisia valikoita. Valikoita rakennettaessa voidaan käyttää erilaisia tyylejä ja näiden avulla tehdä esimerkiksi täysin Windows-ohjelman näköinen Silverligt-sovellus.

Sovelluksen kokonaisuutta voidaan parantaa animaatioilla ja erilaisilla tehosteilla. Näin käyttäjälle on loogisempaa, mitä sovelluksessa oikein tapahtuu. Sen sijaan, että välillä ruudussa on pelkkää tyhjää ja seuraavassa tilanteessa ruudulla on täysin uudenlainen sisältö. Tähän vaikuttaa osaltaan myös sovelluksen suunnittelu, minkälainen sovellus halutaan ja tarvitaan.

Silverlightin avulla voidaan sivustojen yleisilmettä parantaa käyttämällä sivuilla Silverlight-elementtejä perinteisen HTML-koodin osana. Vaikka elementit olisivat täysin samoja kuin vastaavat elementit perinteisessä HTML:ssä, niistä saatava lisäarvo on niiden muokattavuus ja parempi toiminnallisuus.

Napit voidaan muokata sellaisiksi kuin halutaan. Voidaan käyttää monissa elementeissä moniulotteisimpia (esimerkiksi erilaiset liukuvärit) värimaailmoja pelkkien tasaisten värien ja valmiiden taustakuvien sijaan. Tarvittaessa voidaan tukea myös animaatioita ja helpottaa näin ollen käyttäjää havainnoimaan, mikä nimenomainen komponentti on nappi.

Silverlight-teknologian tapaa noutaa dataa tietokannasta voidaan pitää myös käyttäjäkokemusta parantavana. Kuten luvussa 8.3 on kerrottu, toimenpide suoritetaan asynkronisesti. Näin ollen sovellus lataa ja esittää käyttäjälle kaikki elementit sovelluksen pyöriessä jatkuvasti, eikä siinä esiinny koko sovellusta jumittavia lataustaukoja.

Sovelluskehittäjän tulee huomioida asynkroninen tiedonnouto ja informoida asia jotenkin sovelluksen käyttäjälle. Käyttäjän tulee saada riittävän informatiivinen tieto siitä, että tietoa vielä noudetaan, erityisesti jos noutamisessa kestää odotettua kauemmin. Sovelluskehittäjä voi esimerkiksi esittää elementissä latauspalkin, jota päivitetään sitä mukaa kun tietoa saadaan noudettua.

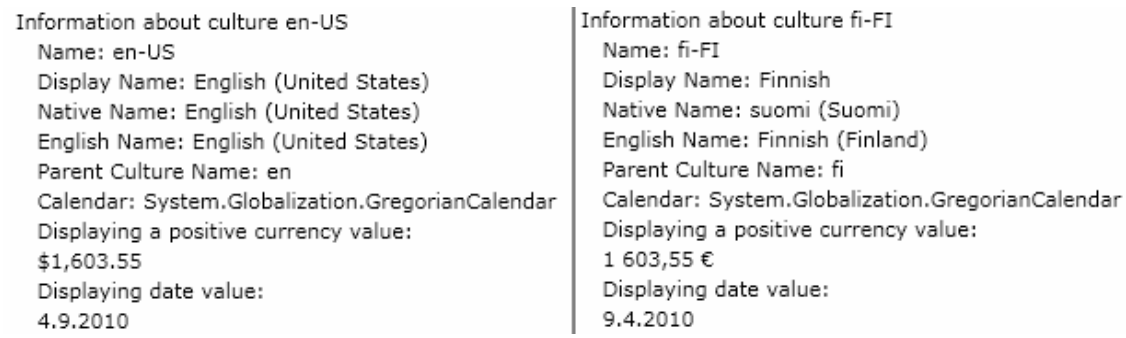
Kun tieto on noudettu ja se on käsitelty sellaiseen muotoon kuin se halutaan sovelluksessa esittää, se voidaan sitoa elementtiin. Tässä yhteydessä Silverlight hahmontaa komponentin uudestaan, ja se on käytettävissä noudetulla tiedolla. Tiedon sitominen on äärimmäisen tehokas ja mahdollisuus kaksisuuntaisuuteen helpottaa monen asian tekemistä. Tiedon sitomisesta on kerrottu luvussa 8.4.

Silverlightin lokalisointimalli (kerrottu enemmän luvussa 8.1) tuo myös lisäarvoa sovellukselle ja parantaa etenkin käyttäjäkokemusta. Käyttäjän ei tarvitse miettiä, missä muodossa tiedot syötetään, kun sovelluksessa on huolehdittu, että kulttuuriasetukset ovat kunnossa. Kuvasta 20 voidaan nähdä kuinka tekstikenttään täytetään tietoa kulttuuriasetuksia hyödyntäen.

```
...
decimal posValue = 1603.5471m;
TextBlock tb = new TextBlock();
CultureInfo cCulture = CultureInfo.CurrentCulture;
//CultureInfo cCulture = new CultureInfo("en-US");
tb.Text += "Information about culture " + cCulture.Name + "\n";
tb.Text += String.Format("  Name: {0}\n", cCulture.Name);
tb.Text += String.Format("  Display Name: {0}\n",
cCulture.DisplayName);
tb.Text += String.Format("  Native Name: {0}\n", cCulture.NativeName);
tb.Text += String.Format("  English Name: {0}\n",
cCulture.EnglishName);
tb.Text += String.Format("  Parent Culture Name: {0}\n",
cCulture.Parent.Name);

tb.Text += String.Format("  Calendar: {0}\n",
cCulture.Calendar.ToString());

tb.Text += "Displaying a positive currency value:\n";
tb.Text += posValue.ToString("C", cCulture);
...
```

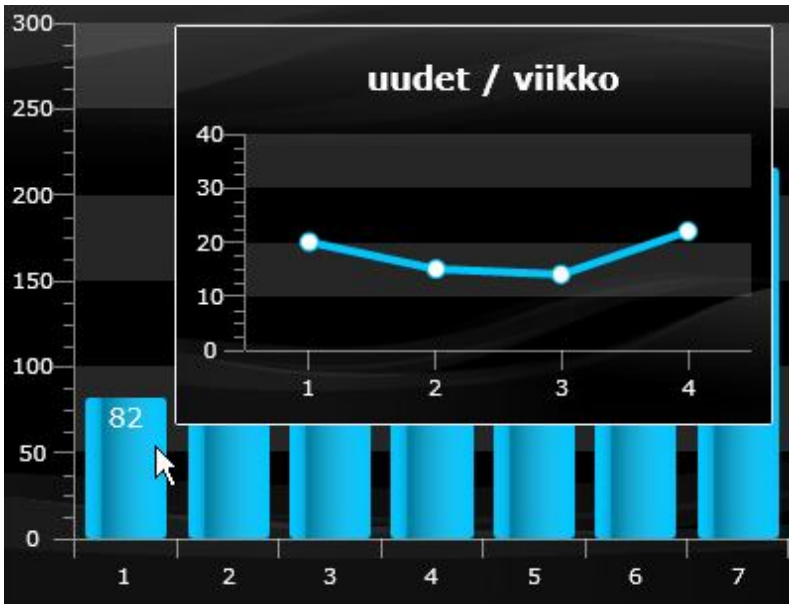


Kuva 20. Tietoja ajettuna Silverlight-sovelluksen läpi, jossa toisella kerralla on vaihdettu kulttuuriasetukset.

Samoin esitettävä tieto on helppolukuisempaa, kun ei tarvitse miettiä, missä kohtaa esitettyä päivämäärää on kuukausi ja päivä, valuuttatiedoissa on tarvittavat tuhat- ja desimaalierottimet paikallaan ja valuuttaa osoittava symboli käyttäjälle tutussa paikassa.

Vain kotimaisilla kielillä käytettävissä sovelluksissa tämä ei ole niin suuressa arvossa kuin sovelluksissa, joissa joudutaan tukemaan myös muiden kulttuurien esitystapoja.

Silverlight-maailman tietynasteinen vapaus mahdollistaa myös informatiivisemman sisällöntuoton, joka parantaa edelleen käyttäjäkokemusta, tuo näyttävyyttä ja helpottaa käyttöä. Tästä hyvänä esimerkkinä on perinteisen työkaluohjeen (tooltip) muokkaus.



Kuva 21. Kaavion pylvään työkaluohjeessa näytetään uusi kaavio.

Kuten kuvasta 21 voidaan nähdä, sovelluksessa esitetään tietoa palkkidiagrammin avulla. Yksi palkki vastaa yhtä vuoden kahdestatoista kuukaudesta. Käyttäjän liikuttaessa hiiren kohdistimen palkin päälle, avautuu työkaluohje, jossa voidaan esittää vastaavanlainen diagrammi. Tässä esimerkissä työkaluohjeessa käytetään viivadiagrammia kuvaamaan kyseisen kuukauden luvun muodostumista kyseisen kuukauden viikkojen luvuista.

Vaikka työkaluohje on hyvin muokattava Silverlight-maailmassa, on olemassa yksi rajoite. Työkaluohjeeseen ei voida sijoittaa komponentteja, joita käyttäjä voisi painella tai jotka vaatisivat käyttäjän toimia (napit, kalenterit jne.).

Prototyyppiä (kerrottu enemmän luvussa 11) suunniteltaessa ja esiversioita testattaessa huomioitiin myös Silverlightin toimivuus erilaisissa kohdistus tilanteissa, joissa vanhoilla tekniikoilla on useasti ollut parannettavaa. Hyvänä esimerkkinä tästä oli sovellukseen upotettu DataGrid, jossa esitettiin tietoja tietokannan taulusta.

DataGridiin tuotiin tietoa niin paljon, että sitä ei mahtunut esittämään sille varatussa tilassa kokonaan. Kun käyttäjä halusi vierittää ikkunan sisältöä alaspäin, tilanteessa riitti pelkkä hiiren siirtäminen elementin päälle ja tämän jälkeen hiiren rullan pyörittäminen. Ei vaadittu ylimääräisiä napsautuksia elementtiin, eikä hiiren kohdistinta tarvinnut tähdätä DataGridin sivulla näkyvän vierityspalkin päälle.

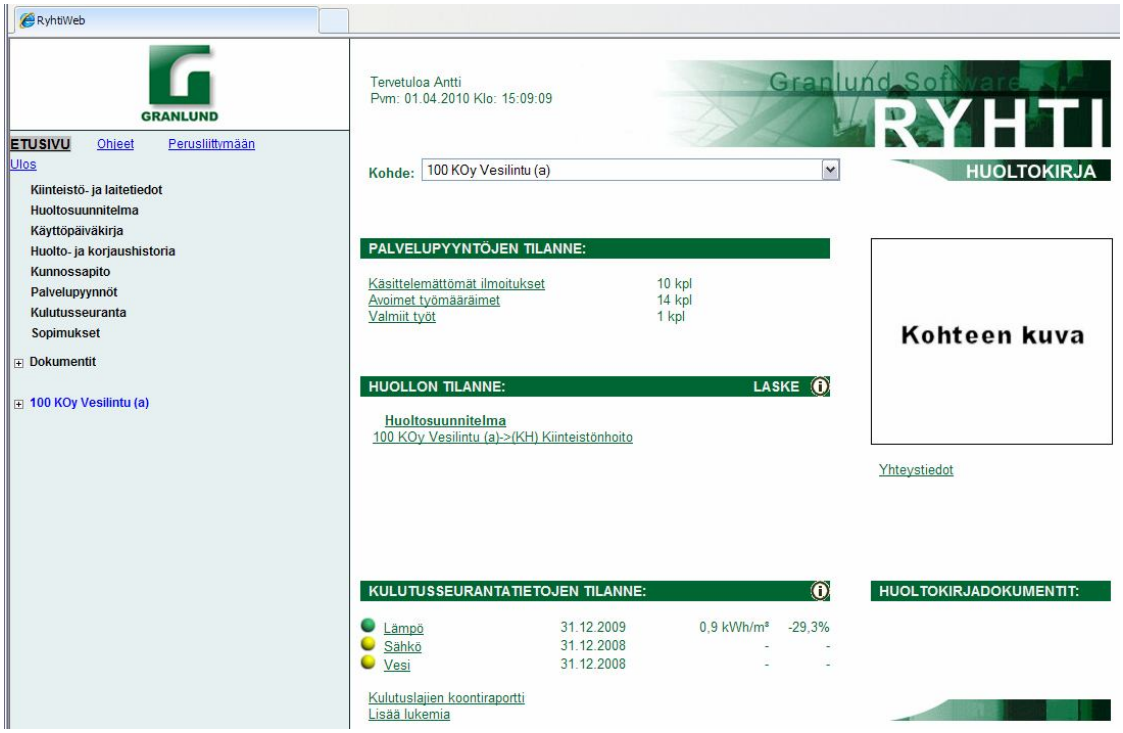
Edellä kuvattu tilanne vaatii tietysti, että Silverlight-elementti on ensin saanut kohdistuksen (mutta tämä ei eroa HTML-sivusta, koska silloinhan kohdistus on jo selaimella, jossa internet-sivu esitetään). Ongelma voi esiintyä etenkin, jos Silverlight-komponentteja käytetään perinteisen HTML-komponenttien seassa tai vaihtoehtoisesti Silverlight-sovellusta osana perinteistä HTML-sivua.

Silverlightin tarjoaman lisäarvon ansiosta se on onnistunut saamaan jalansijaa internet-sovellusten maailmassa kilpailijoiltaan. Osoituksena tästä teknologiaa on käytetty muun muassa suurten tapahtumien internet-lähetyksissä (muun muassa Wimbledon 2009 ja Vancouverin olympialaiset 2010).

11 Ryhti NG

RYHTI™ on tietokantapohjainen ylläpidon hallintajärjestelmä, jolla voidaan johtaa kiinteistöjen tai tuotannollisten järjestelmien ylläpitoon liittyviä tarpeita (12).

Ryhti NG on prototyyppi, jonka avulla haluttiin tutkia ja toteuttaa Silverlightin näkökulma eräänlaisesta sovelluksen yhteenvetosivusta.



Kuva 22. Ryhdin huoltokirja.

Yhteenvetosivulla käyttäjälle esitetään tärkeimpiä lukuja ja tietoja järjestelmän prosesseista, kuten kuvasta 22 voidaan havaita. Yhteenvetosivulla käyttäjälle on myös mahdollista vaihtaa tutkittavana olevaa tuotannollista järjestelmää.

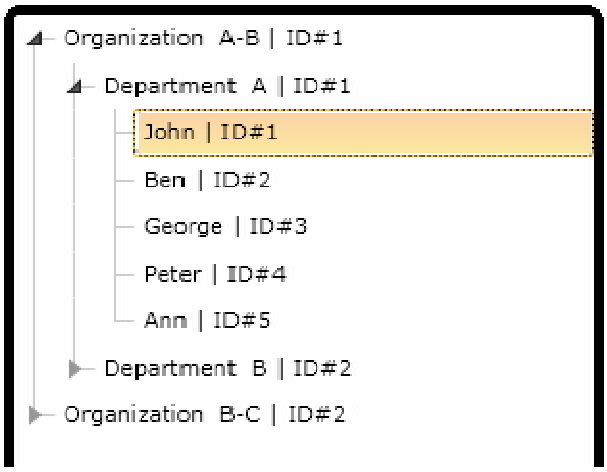
Prototyyppiä varten tutkittiin erilaisia Silverlight-demoja ja -tutoriaaleja, joiden avulla Silverlight-maailma alkoi avautua. Projektin aloitushetkestä alkaen oli tiedossa, että Telerik tarjoaa laadukkaita lisäkomponentteja Silverlight-alustalle nk. RadControlsien muodossa. Tarkoituksena olikin selvittää, kuinka paljon näistä komponenteista saa irti ja miten paljon ne ovat muokattavissa omiin käyttötarpeisiin.

Aluksi prototyyppiin lähettiin suunnittelemaan ja toteuttamaan navigointia, jolla päästäisiin valitsemaan tuotannollinen järjestelmä. Tämän jälkeen vuoroon tulisi varsinaisten yhteenvetotietojen esittäminen.

11.1 Tuotannollisten järjestelmien navigointi

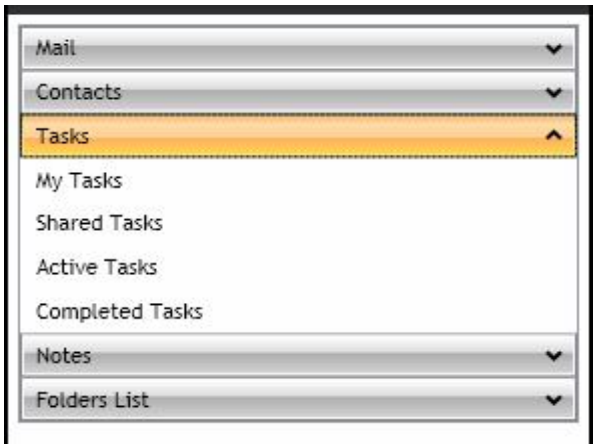
Internet-sovelluksien maailmassa on suunniteltu ja toteutettu monenlaisia navigointeja. Toiset ovat toimivia ja toiset vähemmän toimivia ratkaisuja. Silverlightin avulla haluttiin toteuttaa jotain aivan uutta.

Tuotannolliset järjestelmät muodostavat kolmitasoisen hierarkian, näin ollen navigoinnin avulla piti pystyä esittämään tietoa kolmella eri tasolla. Ensimmäinen mieleen tuleva vaihtoehto on perinteinen puumalli, jossa käyttäjä navigoi. Hyvin perinteinen puumalli on esitetty kuvassa 23.



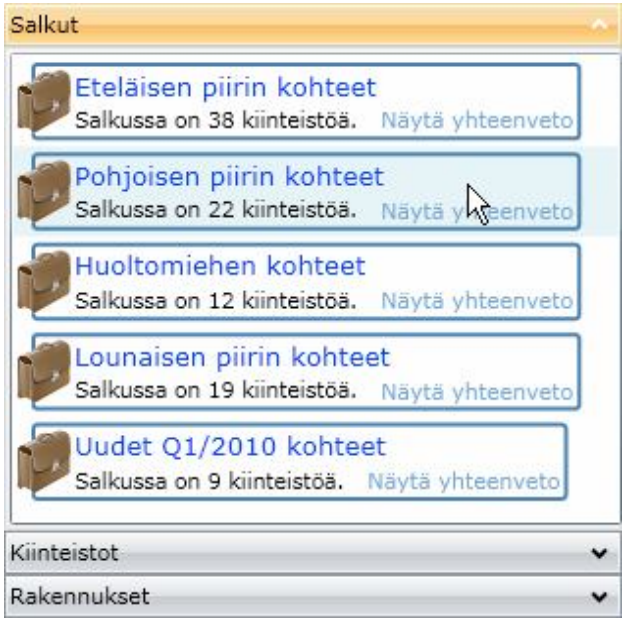
Kuva 23. Perinteinen puunäkymä (RadTreeView).

Perinteinen puumalli oli kuitenkin jotain liian perinteistä, joten tutkittiin vaihtoehtoisia ratkaisuja. Telerikin Silverlightille tarjoamasta ”RadControls for Silverlight”-kirjastosta löytyi komponentti RadPanelBar.



Kuva 24. RadPanelBar, navigointielementti. (8)

RadPanelBar on navigointielementti, jonka avulla voidaan ryhmitellä loogisia kokonaisuuksia, kuten kuvasta 24 voidaan havaita. Elementti toimii joko siten, että siitä laajentuu yksi kokonaisuus kerrallaan (sulkien aina avoimen), tai vaihtoehtoisesti voidaan määritellä, että monta kokonaisuutta voi olla laajennettuna samaan aikaan.



Kuva 25. Ryhti NG, navigointi avoinna ensimmäiseltä tasolta.

Navigoinnin ensimmäinen taso toteutettiin perinteisen listakontrollin avulla. Käyttäjän selatessa järjestelmiä hän voi suodattaa toisen tason vaihtoehtoja tekemällä valinnan ylemmällä tasolla. Listakontrollia ehostettiin tekemällä siihen oma malline, jonka avulla pystytään esittämään lisäinformaatiota käyttäjälle. Navigoinnin ensimmäinen taso on nähtävissä kuvasta 25.

Käyttäjän toimiessa pääsääntöisesti hierarkiamallin toiselle tasolla haluttiin siihen panostaa toteuttamalla jotain ainutlaatuista Silverlightin avulla. Ratkaisua varten tutustuttiin Telerikin RadControlssista löytyvään RadCoverFlow-komponenttiin. Komponentti on alun perin suunniteltu valokuvien esittämistä varten. Kuvassa 26 on esitetty kyseinen komponentti toiminnassa.

Kuva 26. Telerik RadCoverFlow toiminnassa. (8)

Komponenttia tutkittiin aluksi paljon, mahdollistaako se muun sisällön kuin valokuvien esittämisen ja minkälaista kuormitusta komponentti aiheuttaa järjestelmälle, jossa sitä pyöritetään, kun selattavissa on ääritilanteessa toistasataa vaihtoehtoa.

Hyvin äkkiä selvisi, kiitos yleisen Silverlightin mallin, että RadCoverFlow'n yksittäiseen elementtiin voidaan helposti sijoittaa muutakin sisältöä kuin valokuvia. Toteutuksen pohjaksi valittiin Grid-komponentti, johon sijoitetaan valokuva sekä tekstiä ja nappi.

Kuva 27. RadCoverFlow'n elementti, jossa pohjalla valokuva, oikeassa yläkulmassa linkkipainike ja alareunassa lisätietoja.

Tekstin avulla voidaan näin esittää järjestelmän nimi valokuvan päällä. Käyttäjällä on näin ollen kaksi tapaa muistaa haluamansa järjestelmä, joko kuvamuistilla tai nimen perusteella. Nappia painamalla käyttäjällä on mahdollisuus saada lisätietoja valitusta järjestelmästä. Kuvassa 27 on esitetty yhtä järjestelmää kuvaava elementti.

Kuva 28. Valitun järjestelmän lisätietojen näyttäminen.

Lisätiedot näytetään animaatiotehosteen avulla. Animaatio on esitetty kuvassa 28. Sisältö luodaan sovelluksen taustakoodissa, jolloin se ladataan vain käyttäjän halutessa eikä vie turhaan resursseja.

Seuraavaksi vaakasuuntainen selausmalli haluttiin muuttaa pystysuuntaiseksi. Tämä oli alkumetreillä aiheuttaa ongelmia, kunnes selvisi, että Telerik oli uusimmassa RadControls-kirjastossaan mahdollistanut komponentin käytön myös pystysuuntaisena. Parametrin avulla komponentti voitiin kääntää pystyasentoon. Tämä toisaalta myös pakottaa pois kuvassa 26 näkyvän peilausefektin. Pystysuuntainen käyttö ei ole kontrollin puolesta ihan niin tehokkaasti toteutettu kuin vaakamalli, mikä alkoi näkyä pienenä hitautena.

Pystysuuntainen navigaatio toimii siten, että keskellä komponenttia on valittuna oleva järjestelmä ja muut järjestelmät sijoittuvat valitun järjestelmän ylä- ja alapuolelle. Tämä on havaittavissa kuvassa 28 animaatioketjun ensimmäisessä kuvassa. Valittua järjestelmää voi vaihtaa napsauttamalla kuvassa näkyvää toista järjestelmää tai vaihtoehtoisesti käyttämällä hiiren rullaa tai komponentin sivuun sijoitettua liukupalkkia. Valintaa vaihdettaessa järjestelmät siirtyvät liukuanimaatiolla paikoilleen.

Pystysuuntaista navigaatiota tuli seuraavaksi hieman kehittää. Tarkoituksena oli mahdollistaa suurempien kokonaisuuksien esittäminen, myös hieman heikko-tehoisemmalla koneella.

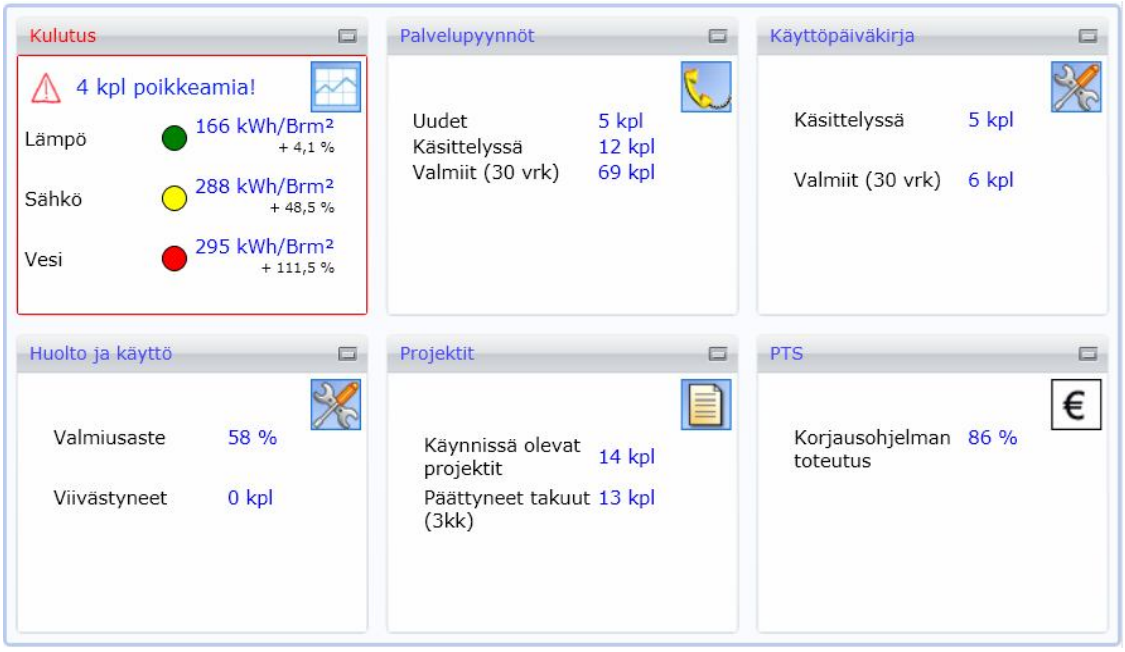
Silverlightin tapa hahmontaa asioita oli kohta, josta lähdettiin hakemaan resurssien säästöä. Tutkimalla RadCoverFlow’n toimintaa, voitiin havaita, että navigaatiosta käyttäjän näkymättömiin liukuvat elementit olivat Silverlightin mielestä aina näkyvillä, ja näin ollen Silverlight hahmonsi jokaista navigaatioon lisättyä elementtiä, oli niitä sitten viisi tai sataviisi.

Pystysuunnassa käyttäjä pystyi hahmottamaan nykyisellä toteutuksella seitsemän elementtiä, valitun elementin ja tämän yläpuolella olevat kolme sekä valitun alapuolelle jäävät kolme elementtiä. Näin ollen komponentin kylkeen tehtiin laajennus, joka huolehtii käyttäjän silmistä poissa olevien komponenttien oikeasta piilottamisesta (tästä puhuttiin luvussa 8.6). Tämän jälkeen komponentti keventyi huomattavasti ja sitä voitiin ajatella käytettäväksi toisen kokonaisuuden rinnalla.

11.2 Yhteenvetotiedot

Yhteenvetotiedoille oli alusta asti yksi merkittävä vaatimus. Yhteenvetotietoja tulisi olemaan useita, ja näistä haluttaisiin näyttää erilaisia tietoja käyttäjälle. Lisäksi yhteenvetotiedoista tulisi pystyä näkemään laajennettu ja kutistettu näkymä.

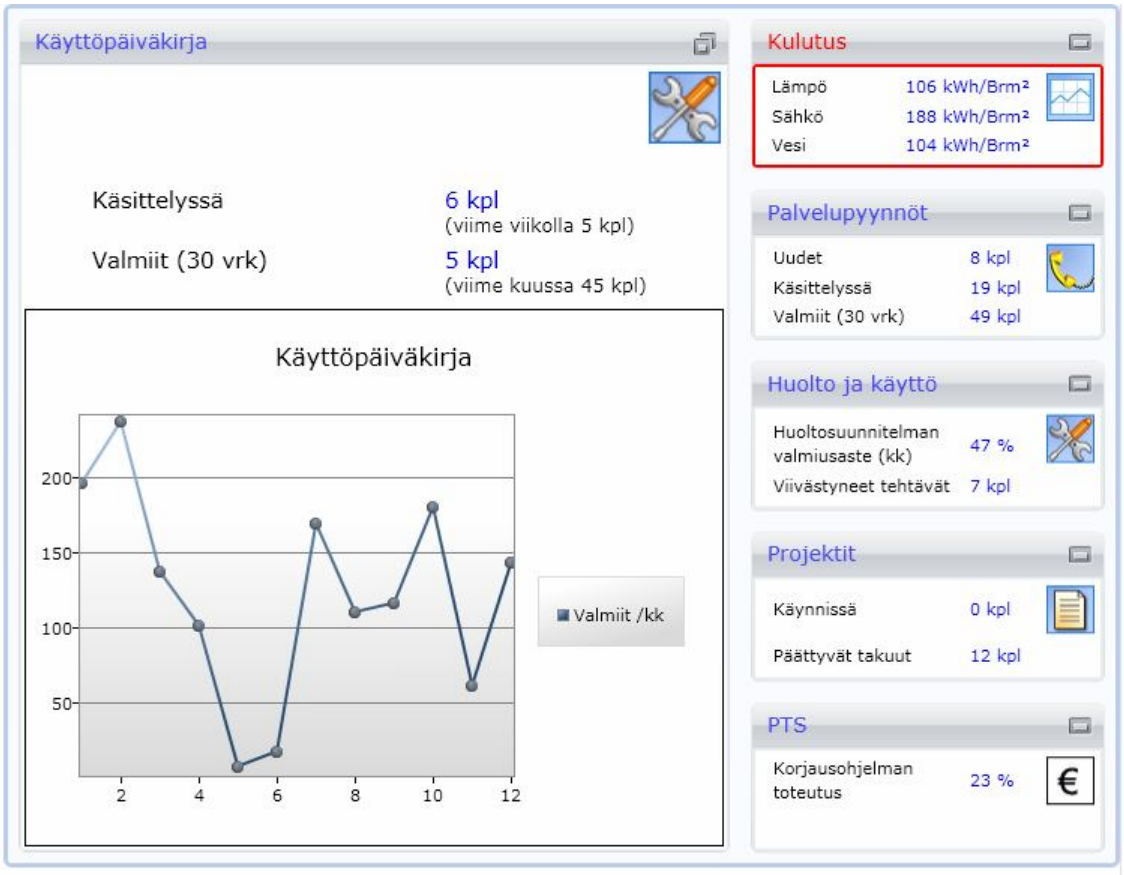
Telerikin komponentteja tutkittaessa jossain vaiheessa vastaan tuli RadTileView-komponentti, jossa käytetään lisänä RadFluidContentControlia. Tämä mahdollistaa useamman selkeästi erilaisen kokonaisuuden esittämisen. Jokainen kokonaisuus ”elää” omassa ikkunassaan.



Kuva 29. RadTileView, jossa jokainen ikkuna on samanarvoisessa tilanteessa.

Ikkunat voivat olla kaikki samanarvoisia keskenään, kuten kuvassa 29, tai vaihtoehtoisesti voidaan suurentaa yksi ikkuna, jolloin muut ikkunat pienenevät komponentin laidoille. On ohjelmoijan täydessä kontrollissa, mille reunalle ja minkälaisina muut ikkunat siirtyvät.

Animoinnin avulla komponentti tarjoaa myös paljon visuaalista silmäniloa. Ikkunan koon ja sisällön muutokseen on tuettu animaatioita, ja näin ollen ne ikään kuin liukuvat paikalleen. Käyttäjä voi myös halutessaan järjestellä ikkunat haluamilleen paikoille.



Kuva 30. RadTileView, jossa yksi ikkuna on suurennettuna ja muut ikkunat pienennettynä komponentin oikeaan reunaan.

RadFluidContentControl mahdollistaa erilaisen sisällön esittämisen ikkunassa sen eri olomuodoissa (suuri, normaali ja pieni). Näin ollen suurennetussa ikkunassa esitetään erilaisia kaavioita ja kuvia riippuen siitä, mitä kokonaisuutta se edustaa. Tasavertaisessa tilanteessa (kaikki ikkunat tilassa normaali) ikkunat sisältävät kokonaisuuden perustiedot. Ikkunoiden ollessa pienennetyssä tilassa niissä esitetään vain tärkein tieto. Molemmat tilat on havaittavissa kuvasta 30, suuri tila käyttöpäiväkirja ikkunasta ja muut ikkunat on pienennetyssä tilassa.

11.3 Tulevaisuus

Prototyyppiä on esitetty sen valmistuttua (helmikuussa 2010), ja se on saanut positiivisen vastaanoton. Prototyypistä ei sinällään ole jatkon kannalta valmista kuin osa komponenteista ja pieni osa sisällöstä. Prototyyppi kun tukeutui täysin nk. hölynpölytietoihin. Valmiin prototyypin kuvakaappaus on esitetty liitteessä 5.

Varsinaiseen sovellukseen siirryttäessä on tiedostettu, että kaiken tiedon noutaminen tietokannasta, joka halutaan sovelluksessa esittää, on oma melkoisen iso projektinsa. Lisäksi prototyypissä keskityttiin olennaisiin toimintoihin ja se on näin ollen täysin tyylittelemätön.

Prototyypin jatkokehitys noussee esille tulevaisuudessa. Prototyypistä on mietitty mahdollista vaihtoehtoista tapaa käyttää sovellusta. Näin ollen käyttäjiä ei sidota uuteen teknologiaan, vaan tarjotaan uusi ja vaihtoehtoinen tapa tehdä asioita.

12 Yhteenveto

Insinöörityön tavoitteena oli tutustua Microsoftin Silverlight-teknologiaan, sen vaatimuksiin ja rajoituksiin ohjelmistokehityksessä. Teoriaosuudessa tutustuttiin pintapuolisesti internet-sovellusten kehityskaareen, jonka jälkeen tutustuttiin tarkemmin Silverlight-teknologiaan. Teoriaosuus mahdollisti insinöörityön konkreettisemman vaiheen eli Silverlight-teknologialla toteutettavan prototyypin toteutuksen.

Prototyypin toteutuksesta Olof Granlund Oy:ssä oltiin tyytyväisiä. Projektin alussa annetut tavoitteet täytettiin tältä osin ja yhtiössä ollaan hyvin kiinnostuneita prototyypin jatkokehityksestä tulevaisuudessa.

Ryhti NG -prototyypistä on tarkoitus kehittää varteenotettava vaihtoehto perinteisen yhteenvetosivun rinnalle. Silverlight-teknologia osoittautui helposti omaksuttavaksi, kun keskityttiin jo tarjolla oleviin komponentteihin ja näiden jatkokehitykseen. Prototyypin toteutuksessa pystyttiin toteuttamaan suunniteltu käyttöliittymä, eikä mahdottomia ongelmia esiintynyt.

Insinöörityö onnistui täyttämään sille asetetut tavoitteet. Työn merkittävimpänä antina on Silverlight-teknologian tuntemus, sen tarjoamat mahdollisuudet ja huomiot rajoituksien suhteen. Työ osoittaa, että Microsoft Silverlight on kehittynyt paljon ja kehitystyötä jatketaan edelleen. Tekniikka on noussut harkinnan arvoiseksi sovelluskehityksessä.

Lähteet

1 Microsoft Silverlight. (WWW-dokumentti.) Wikipedia the free Encyclopedia. <http://en.wikipedia.org/wiki/Microsoft_Silverlight>. Luettu 30.12.2009.

2 Mitä tarkoittaa RIA? (WWW-dokumentti.) Uoma Oy. <http://www.uoma.fi/ajankohtaista/artikkelit/20090325-ria.html> Luettu 22.12.2009.

3 MacDonald, Matthew. Pro Silverlight 3 in C#. New York: Apress, 2009.

4 MSDN - Silverlight. (WWW-dokumentti). Microsoft, Developer Network. <http://msdn.microsoft.com/en-us/library/cc838158(VS.95).aspx>. Luettu 9.12.2009.

5 Configuring a Web Server to Host Silverlight Content. (WWW-dokumentti). Tim Sneath. <http://blogs.msdn.com/tims/archive/2008/03/18/configuring-a-web-server-to-host-silverlight-content.aspx> Kirjoitettu 18.3.2008, luettu 15.12.2009.

6 Silverlight 4 Beta Information. (WWW-dokumentti). Microsoft Silverlight. <http://silverlight.net/getstarted/silverlight-4-beta/> Luettu 28.12.2009.

7 Text and Fonts. (WWW-dokumentti). Microsoft Silverlight Developer Center. <http://msdn.microsoft.com/en-us/library/cc189010%28VS.95%29.aspx> Luettu 5.1.2010.

8 Silverlight Demos. (WWW-dokumentti.) Telerik. <http://demos.telerik.com/silverlight/> Luettu 22.12.2009.

9 HTML. (WWW-dokumentti.) Wikipedia the free Encyclopedia. <http://fi.wikipedia.org/wiki/HTML>. Luettu 6.3.2010.

10 ViewModel Pattern in Silverlight. (WWW-dokumentti). Nikhil Kothari. <http://www.nikhilk.net/Silverlight-ViewModel-Pattern.aspx> Luettu 15.1.2010

11 Silverlight Architecture. (WWW-dokumentti). Microsoft Developer Network <http://msdn.microsoft.com/en-us/library/bb404713(VS.95).aspx>. Luettu 16.3.2010

12 RYHTI™. (WWW-dokumentti). Insinööritoimisto Olof Granlund Oy – RYHTI. <www.ryhti.net/ryhti/>. Luettu 19.3.2010.

13 Optimizing Silverlight with Enable Redraw Regions. (WWW-dokumentti.) Gavin WIgnallin blogi. <http://www.silverlightbuzz.com/2009/11/17/optimizing-silverlight-with-enable-redraw-regions/>. Luettu 27.1.2010.

14 MSDN - Deep Zoom. (WWW-dokumentti). Microsoft Developer Network. <http://msdn.microsoft.com/en-us/library/cc645050(VS.95).aspx>. Luettu 27.3.2010.

15 Using Local Connection. (WWW-dokumentti). Switch On The Code. <http://www.switchonthecode.com/tutorials/silverlight-3-using-local-connections>. Luettu 29.12.2010.

16 Displaying SQL Database in a DataGrid using LINQ and WCF. (WWW-dokumentti). Microsoft Silverlight. <http://www.silverlight.net/learn/tutorials/sqldatagrid-cs/>. Luettu 16.12.2010.

17 Silverlight 3 - Using Local Connections. (WWW-dokumentti.) Switch On The Code. <http://www.switchonthecode.com/tutorials/silverlight-3-using-local-connections>. Luettu 16.12.2010.

18 Securing Silverlight Application and WCF Service using ASP.NET Authentication Techniques. (WWW-dokumentti). Mehroz's Experiments blogi. < http://smehrozalam.wordpress.com/2009/01/07/securing-silverlight-application-and-wcf-service-using-aspnet-authentication-techniques/>. Luettu 30.3.2010.

19 MSDN WCF Transport Security Overview. (WWW-dokumentti.) Microsoft, Developer Network. <http://msdn.microsoft.com/en-us/library/ms729700.aspx>. Luettu 30.3.2010.

20 Silverlight Overview. (WWW-dokumentti.) Microsoft Silverlight. <http://www.silverlight.net/getstarted/overview.aspx>. Luettu 30.12.2009.

Liite 1: Kirjautumislomakkeen XAML

```
...
    <Grid x:Name="LayoutRoot" Background="White">
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="130"></RowDefinition>
            <RowDefinition Height="30"></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>

        <Canvas x:Name="canvas" Width="265" Margin="10" Grid.Row="1"
        Background="White">

               <Canvas.Projection>
                      <PlaneProjection/>
               </Canvas.Projection>

            <Grid Background="White" Canvas.Left="10" Canvas.Top="-8"
            Canvas.ZIndex="99" ShowGridLines="True">

                <TextBlock Margin="4, 0" FontSize="12">Log In
                </TextBlock>

            </Grid>

            <Border BorderBrush="Gray" BorderThickness="1"
            CornerRadius="6">

                <Grid x:Name="InnerLayoutRoot" Margin="10"
                Background="White">

                    <Grid.RowDefinitions>
                        <RowDefinition></RowDefinition>
                        <RowDefinition></RowDefinition>
                        <RowDefinition></RowDefinition>
                    </Grid.RowDefinitions>

                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="80"/>
                        <ColumnDefinition />
                    </Grid.ColumnDefinitions>

                    <TextBlock Text="User Name:" Grid.Row="0"
                    Grid.Column="0" Margin="0, 5" FontSize="12" />

                    <TextBox x:Name="UserNameTextbox" Grid.Row="0"
                    Grid.Column="1" Margin="0, 5" Text="" Width="160"
                    FontSize="12" />

                    <TextBlock Text="Password:" Grid.Row="1"
                    Grid.Column="0" Margin="0, 5" FontSize="12" />
```

Liite 1: Kirjautumislomakkeen XAML

```
                <PasswordBox x:Name="PasswordTextbox" Grid.Row="1"
                Grid.Column="1" Margin="0, 5" Password=""
                Width="160" FontSize="12" />

                <Button x:Name="LoginButton" Grid.Row="2"
                Grid.Column="1" Margin="0, 5" Width="80"
                HorizontalAlignment="Left" Content="Log In"
                Click="LoginButton_Click" />

                <Button x:Name="FS" Grid.Row="2" Grid.Column="1"
                Margin="0, 5" Width="80"
                HorizontalAlignment="right" Content="FS"
                Click="FS_Click" />

            </Grid>

        </Border>

    </Canvas>

</Grid>
  ...
```

Liite 2: Silverlightin tukemat latinalaisen tekstin fontit

- Arial
- Arial Black
- Arial Unicode MS
- Calibri
- Cambria
- Cambria Math
- Comic Sans MS
- Candara
- Consolas
- Constantia
- Corbel
- Courier New
- Georgia
- Lucida Grande/Lucida Sans Unicode
- Segoe UI
- Symbol
- Tahoma
- Times New Roman
- Trebuchet MS
- Verdana
- Wingdings
- Wingdings 2
- Wingdings 3

(7)

Liite 3: Yksinkertainen HTML sivu

```
<HTML>
 <HEAD>
  <TITLE>Yksinkertainen HTML-sivu</TITLE>
 </HEAD

 <BODY>

  <TABLE BORDER="1" WIDTH="100%">
   <TR>
    <TD COLSPAN="2">
     <H1>Tervetuloa yksinkertaiselle html-sivulle</H1>
    </TD>
   </TR>
   <TR>
    <TD VALIGN="TOP" WIDTH="30%">
          Viittauksia:<BR>
      <A HREF="#1">Dokumentti 1</A><BR>
      <A HREF="#2">Dokumentti 2</A><BR>
      <A HREF="#3">Dokumentti 3</A><BR>
    </TD>
    <TD VALIGN="TOP" WIDTH="70%">
     Geneeristä tekstiä mahdolliseen sisältölohkoon. Geneeristä tekstiä mahdolliseen
sisältölohkoon. Geneeristä tekstiä mahdolliseen sisältölohkoon. Geneeristä tekstiä mahdolliseen
sisältölohkoon. Geneeristä tekstiä mahdolliseen sisältölohkoon.
    </TD>
   </TR>

  </TABLE>
 </BODY>
</HTML>
```

Liite 4: Silverlightin tukemien käyttöjärjestelmien ja internet-selainten matriisi

OS/selain	IE 6 SP1	IE 6 SV1 (SP2)	IE 7/ IE 8	Firefox	SeaMonkey	Safari	Opera	Google Chrome
Windows Vista/ Windows 7	N/A	N/A	1.0, 2.0, 3.0	1.0, 2.0, 3.0	1.0, 2.0	1.0, 2.0; NPAPI:lla	Epävirallisesti	2.0, 3.0
Windows XP/2003/Home Server	N/A	1.0, 2.0, 3.0	1.0, 2.0, 3.0	1.0, 2.0, 3.0	2.0 Epäviral-lisesti	1.0, 2.0; NPAPI:lla	Epävirallisesti	2.0, 3.0
Windows 2000	2.0, 3.0	N/A	N/A	2.0 Epäviral-lisesti	N/A	2.0; NPAPI:lla	Suunniteltu	N/A
Windows Phone 7 Series	N/A	N/A	Suun-niteltu	N/A	N/A	N/A	N/A	N/A
S60	Suun-niteltu	N/A	N/A	N/A	N/A	N/A	N/A	N/A
FreeBSD	N/A	N/A	N/A	1.0 "tekeillä"	N/A	N/A	N/A	N/A
Linux	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mac OS 10.4/10.5 PowerPC	N/A	N/A	N/A	1.0	N/A	1.0	Suunniteltu	N/A
Mac OS 10.4/10.5 Intel	N/A	N/A	N/A	1.0, 2.0, 3.0	N/A	1.0, 2.0, 3.0	Suunniteltu	N/A

Liite 5: Kuvakaappaus RYHTI NG -prototyypistä.